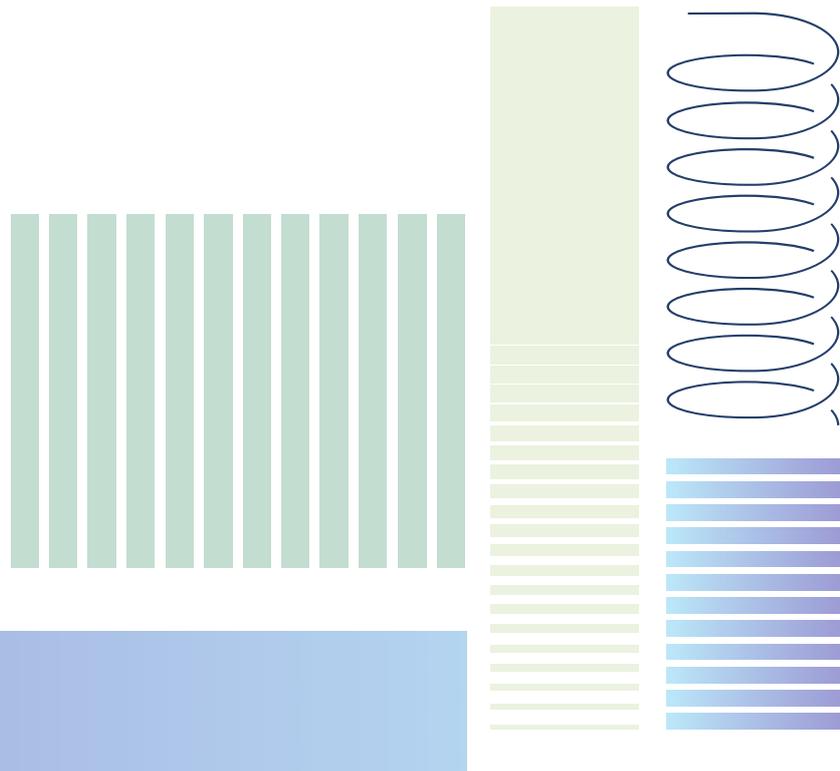


D6.6. FUNCTIONAL EVALUATION REPORT



Deliverable number and name: D6.6. Functional Evaluation Report

Due date: 31/05/2021

Work Package: WP6

Deliverable leader: UVEG

Authors: Cristina Portalés (UVEG), Javier Sevilla (UVEG), Pablo Casanova (UVEG), Thibault Ehrhart (EURECOM), Raphaël Troncy (EURECOM)

Reviewers: Dunja Mladenic (JSI)

Approved by: Jorge Sebastián (UVEG)

Dissemination level: PU

Version: V1.0

Document revision history			
Version	Date	Contributor	Comments
V0.0	04/05/2021	Cristina Portalés	Table of contents
V0.1	15/06/2021	Javier Sevilla	Description of the stress test process
V0.2	16/06/2021	Thibault Ehrhart	Description of the APIS / Queries
V0.3	16/06/2021	Javier Sevilla	Results of the ADASilk APIs stress tests
V0.4	22/06/2021	Javier Sevilla	Result of the thesaurus stress tests
V0.5	25/06/2021	Cristina Portalés, Pablo Casanova	Final testing
V0.6	29/06/2021	Dunja Mladenic	Review suggestions
V0.7	18/07/2021	Raphaël Troncy	Add discussion and conclusion
V0.8	20/07/2021	Daniel Sheerin	English proofreading
V1.0	21/07/2021	Cristina Portalés	Layout



List of acronyms	
AAT	Art and Architecture Thesaurus
API	Application Programming Interface
CPU	Central Processing Unit
ICT	Information and Communication Technologies
JSI	Institut Jozef Stefan
KG	Knowledge Graph
OS	Operating System
RAM	Random Access Memory
SSH	Social Sciences and Humanities
URI	Uniform Resource Identifier
UVEG	Universitat de València Estudi General
WP	Work Package

SILKNOW

Table of contents

1. INTRODUCTION	6
2. SILKNOW'S THESAURUS BROWSER	7
2.1. Description of SILKNOW's Thesaurus.....	7
2.2. Evaluation of the functionalities identified by SSH experts	7
2.2.1. Functionality 01	9
2.2.2. Functionality 02	10
2.2.3. Functionality 03	11
2.2.4. Functionality 04	11
2.2.5. Functionality 05	12
2.2.6. Functionality 06	13
2.2.7. Functionality 07	13
2.2.8. Functionality 08	14
2.2.9. Functionality 09	15
2.2.10. Discussion.....	15
2.3. Stress testing.....	15
2.3.1. Test description.....	16
2.3.2. How to evaluate the result of a request.....	19
2.3.3. Gathered data during the tests	20
2.3.4. SKOSMOS URI	22
2.3.5. SILKNOW PUBLIC API.....	25
2.3.6. SILKNOW SPARQL API	28
2.3.7. Discussion.....	30
3. ADASILK	31
3.1. Description of ADASilk	31
3.2. Evaluation of the functionalities identified by SSH experts	31
3.2.1. Functionality 01	34
3.2.2. Functionality 02	35
3.2.3. Functionality 03	36
3.2.4. Functionality 04	36
3.2.5. Functionality 05	37
3.2.6. Functionality 06	38
3.2.7. Functionality 07	39
3.2.8. Functionality 08	40
3.2.9. Functionality 09	40

SILKNOW

3.2.10.	Functionality 10.....	41
3.2.11.	Functionality 11.....	43
3.2.12.	Functionality 12.....	44
3.2.13.	Functionality 13.....	45
3.2.14.	Functionality 14.....	47
3.2.15.	Functionality 15.....	47
3.2.16.	Functionality 16.....	48
3.2.17.	Functionality 17.....	49
3.2.18.	Functionality 18.....	50
3.2.19.	Functionality 19.....	51
3.2.20.	Functionality 20.....	52
3.2.21.	Functionality 21.....	53
3.2.22.	Functionality 22.....	55
3.2.23.	Functionality 23.....	56
3.2.24.	Functionality 24.....	56
3.2.25.	Functionality 25.....	57
3.2.26.	Functionality 26.....	57
3.2.27.	Discussion.....	58
3.3.	Stress testing.....	58
3.3.1.	Test description.....	58
3.3.2.	Definition of the parameters measured.....	60
3.3.3.	Gathered data during the tests.....	60
3.3.4.	Results for ADASilk Internal API.....	61
3.3.5.	Results for SILKNOW Public API.....	63
3.3.6.	Results for SPARQL API.....	64
3.3.7.	Discussion.....	66
4.	SERVER STATISTICS.....	66
5.	CONCLUSIONS.....	67
	REFERENCES.....	68
	ANNEX I. Request specification for stress test with ADASilk.....	69

This deliverable consists of a test report on the SILKNOW system, mainly developed and integrated in Workpackage 6, and in Workpagage 3. On the one hand, the system includes SILKNOW's Thesaurus, which was defined, designed and implemented in Tasks T3.1, T6.3 and T6.4, and presented in Deliverable D3.1, D6.3 and D6.4, respectively. On the other hand, it includes ADASilk, developed in Task T6.4 and presented in Deliverable D6.4. This deliverable shows the results of a functional and stress testing of both components.

1. INTRODUCTION

This deliverable consists of a technical report describing the functional and stress testing that we have carried out for two of the tools developed in SILKNOW: SILKNOW's Thesaurus Browser (presented in deliverable D3.1) and ADASilk (presented in deliverable D6.4).

Apart from the overall requirements which the tools fulfil, the specific functionalities that should be offered by SILKNOW's Thesaurus Browser and ADASilk were identified by SSH partners in the scope of WP2. They built a table of functionalities for the tools developed in the project (which also includes Virtual Loom and STMaps), and identified which functionalities were relevant to which user profiles (or "personas") and what sectors; these were described as part of D2.4. Therefore, for the functionality evaluation in this deliverable, we review all the identified functionalities one by one, describing where they have been integrated or how they can be used in the current versions of the tools, and providing examples of their use.

While with the functionality evaluation we intend to prove that the required functionalities have been properly integrated into the software tools, with the stress testing we aim to determine their robustness by testing beyond the limits of normal operation. The parameters to be measured in both tools are the same: the response time and the failure rate given a single server configuration. These parameters are measured for both tools in a variety of test scenarios and sets of users. Additionally, each test was repeated five times to obtain mean values.

This deliverable is related to other deliverables in the project, mainly, D2.4, D3.1, D6.3, D6.4, D6.7, D7.3 and D7.6. As mentioned above, in D3.1 "Historical silk multilingual thesaurus", SILKNOW's Thesaurus, which involves dedicated silk heritage terminology, was presented. Its design and implementation were described as part of D6.3 "Ontology web server" and D6.4 "Design and implementation of the multilingual web-based thesaurus". Regarding ADASilk, it was also presented as part of D6.4. It must be noted that the term ADASilk was used after the production of D6.4, where it is referred to as a "web-based search application". A final update

on ADASilk, including the integration of its components, will be included in deliverable D6.7 “System Documentation”. The functionalities of the tools (SILKNOW’s Thesaurus Browser and ADASilk) were established by different consortium partners, mainly those related to the SSH fields, as part of WP2. These functionalities were related to personas and sectors which were defined in D2.4 “Pilot scenario definition”. The usability of ADASilk was evaluated by the general public, including targeted audiences. The results are integrated in D7.3 “Usability evaluation by online users of the system”. In addition, the evaluation of the Thesaurus Browser is presented as part of D7.6 “SILKNOW system evaluation”.

Section 2 of this deliverable is dedicated to the evaluation of SILKNOW’s Thesaurus Browser. It provides an overview of the tool and outlines the functionalities, personas and sectors identified to be relevant for the tool before presenting the stress testing. Section 3 is dedicated to the evaluation of ADASilk and follows a similar structure. Finally, section 4 presents the server statistics and section 5 the conclusions.

2. SILKNOW’S THESAURUS BROWSER

2.1. Description of SILKNOW’s Thesaurus

The multilingual SILKNOW thesaurus is a controlled vocabulary which has a semantic network of unique concepts [1]. Each concept has a unique preferred label per language (657 in English, 658 in Spanish and French and 651 in Italian) as well as multiple alternative labels (or synonyms) in each language. Concepts are arranged into hierarchies, using a Narrower (or its inverse Broader) relationship. Concepts can also have associative concepts using the Related relationship. This thesaurus is unique as it is the only one entirely dedicated to silk heritage, and it includes weaving techniques, materials, depictions and equipment, among other things.

We further developed and deployed the SKOSMOS software [2] in order to search and browse the SILKNOW thesaurus. The so-called SILKNOW Thesaurus Browser is configured to load the data from the SILKNOW Knowledge Graph and generates its views carrying out SPARQL queries to the SILKNOW SPARQL endpoint [3] using the internal SKOSMOS API.

2.2. Evaluation of the functionalities identified by SSH experts

As explained in the introduction, the functionalities that should be offered by SILKNOW’s Thesaurus Browser were identified by SSH partners in the scope of WP2. From the table that they prepared with the description of functionalities, in Table 1 we present those most related to the Thesaurus. In this table, we also give examples and additional comments to better explain the functionalities which the tool should provide, as well as to which personas and sectors this can be of relevance. It is the goal of the functional evaluation to show which of the functionalities were successfully integrated into the Thesaurus Browser and how this integration was achieved. We have added a last column to this table, in grey, which summarizes the results of the functionality evaluation which are presented in the following subsections (2.2.1 to 2.2.9). In order to show if the functionalities were implemented, we have reviewed where they have been integrated into the tool and how they can be accessed and used. We also provide graphical examples for the sake of clarity. Finally, a discussion is given in Section 2.2.10.

SILKNOW

Functionality ID	Description of the functionality	Examples, additional comments	Personas	Sectors	Fulfilled?
<i>General functionality: Having access to a more personalized experience</i>					
01	Making all our web resources responsive to all user devices (desktop/laptop, tablet, smartphone)		All	All	Yes
02	Choosing the language of the Web interface and the thesaurus terms		All	All	Yes
<i>General functionality: Searching</i>					
03	Using a simple search interface	Based on metadata (text-only)	All	All	Yes
04	For queries that provide no answers (or for all queries), suggest related search terms, in the same or other languages	“Related” (search terms) can mean different things.	All	All	Yes
05	Searching the thesaurus for terms and accessing their definition		Museum director	Research and Education	Yes
06	Giving the user the option to choose in the thesaurus the languages of the definitions, related terms and translations of these terms	For instance, letting the user choose whether he only wants definitions in French or also translations into all four languages	Museum director	Research and Education	Yes
07	Browsing visually the thesaurus’ terms, through drop-down hierarchies		All	Research and Education	Yes
<i>General functionality: Finding and visualizing</i>					
08	Linking SILKNOW data to other online resources such as Europeana, Wikidata, IdRef, RAMEAU, Library of Congress authorities... etc.	We mean an automated process, nothing that has to be invoked by the user, but an added level of	ICOM Conservation Committee employee	Cultural heritage and leisure	Yes

		reference provided			
09	Retrieving bibliographic references related to a record, when available.		College Student	Research and Education	Yes

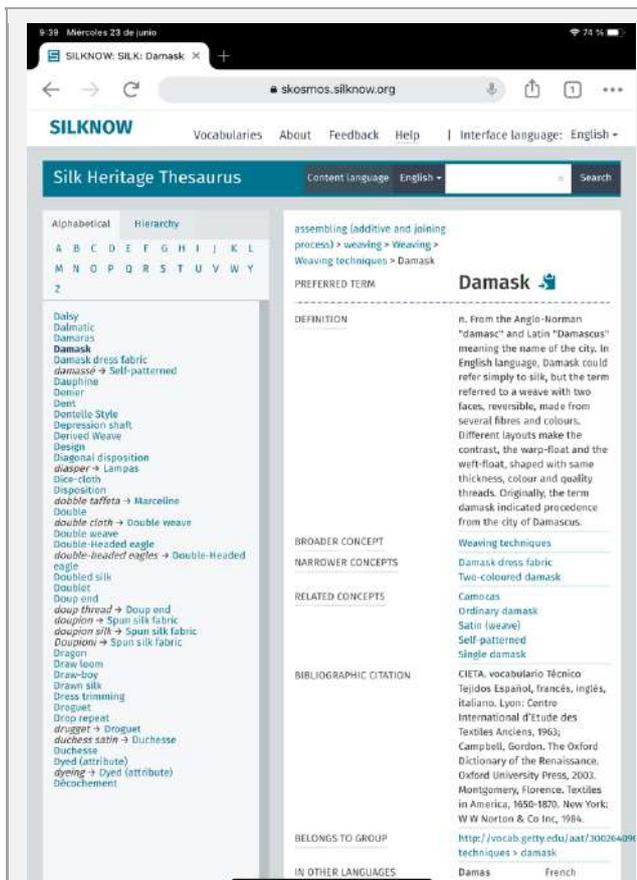
Table 1. List of the functionalities that SILKNOW’s Thesaurus Browser should fulfil according to SSH partners, and their relation to personas and sectors that were identified in D2.4. The last column, in grey, summarizes the outcomes of the functionality evaluation.

2.2.1. Functionality 01

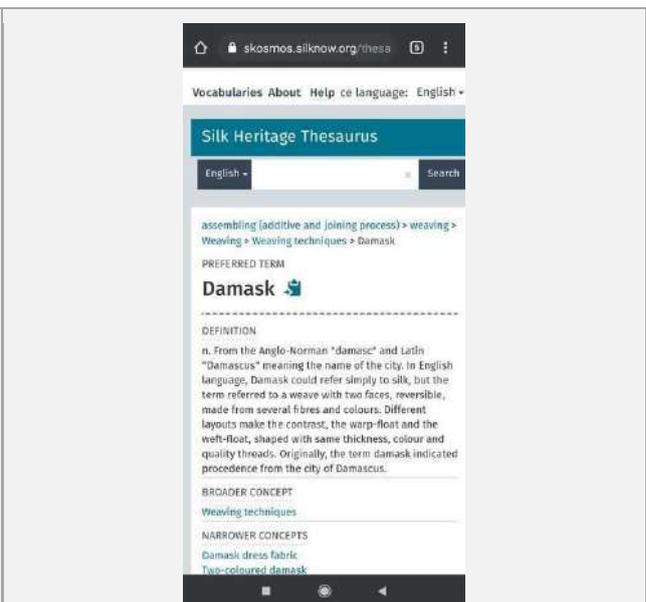
Description of the functionality: Making all our web resources responsive to all user devices (desktop/laptop, tablet, smartphone).

Fulfilled: Yes.

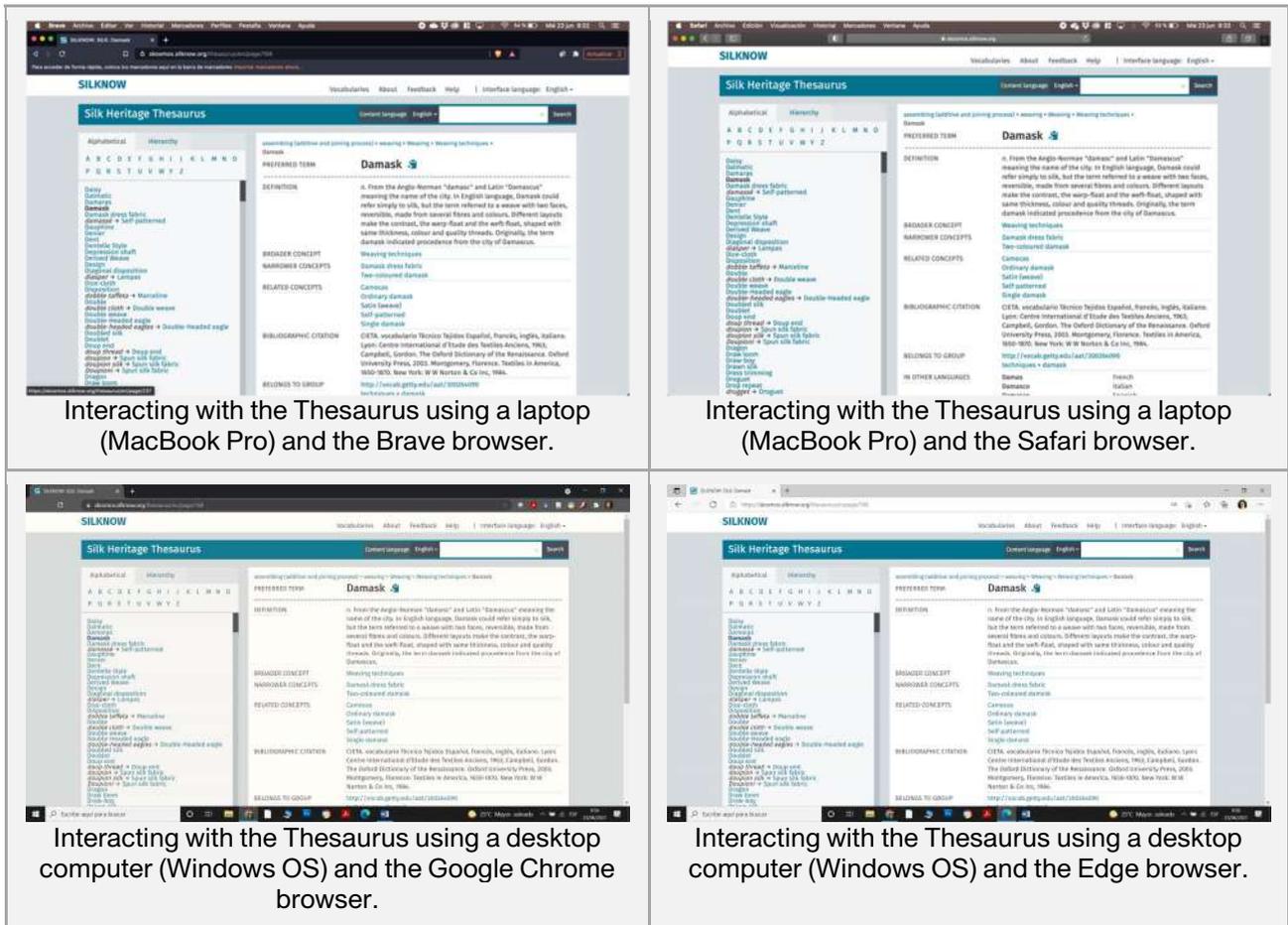
Implementation/use: SILKNOW’s Thesaurus Browser is prepared to be used on standard devices (desktop/laptop, tablet, smartphone) and making use of most modern web browsers (Chrome, Safari, Edge, etc.). To show this, a few examples are provided in Figure 1, where the same action, browsing the term “damask”, is applied using different devices and browsers, also involving different operating systems.



Interacting with the Thesaurus using an iPad and the Google Chrome browser.



Interacting with the Thesaurus using an Android mobile phone, in vertical (top) and horizontal (down) mode, and the Google Chrome browser.



Interacting with the Thesaurus using a laptop (MacBook Pro) and the Brave browser.

Interacting with the Thesaurus using a laptop (MacBook Pro) and the Safari browser.

Interacting with the Thesaurus using a desktop computer (Windows OS) and the Google Chrome browser.

Interacting with the Thesaurus using a desktop computer (Windows OS) and the Edge browser.

Figure 1. Screenshots of SILKNOW's Thesaurus Browser to test functionality 01. Different devices and browsers were used.

2.2.2. Functionality 02

Description of the functionality: Choosing the language of the Web interface and the thesaurus terms.

Fulfilled: Yes.

Implementation/use: At the top of the web page there is a selector to choose the language for the user interface. In the search box there is also another selector to change the language the user wants to search terms for (there is the option to search for all the languages at the same time). This is illustrated in Figure 2.

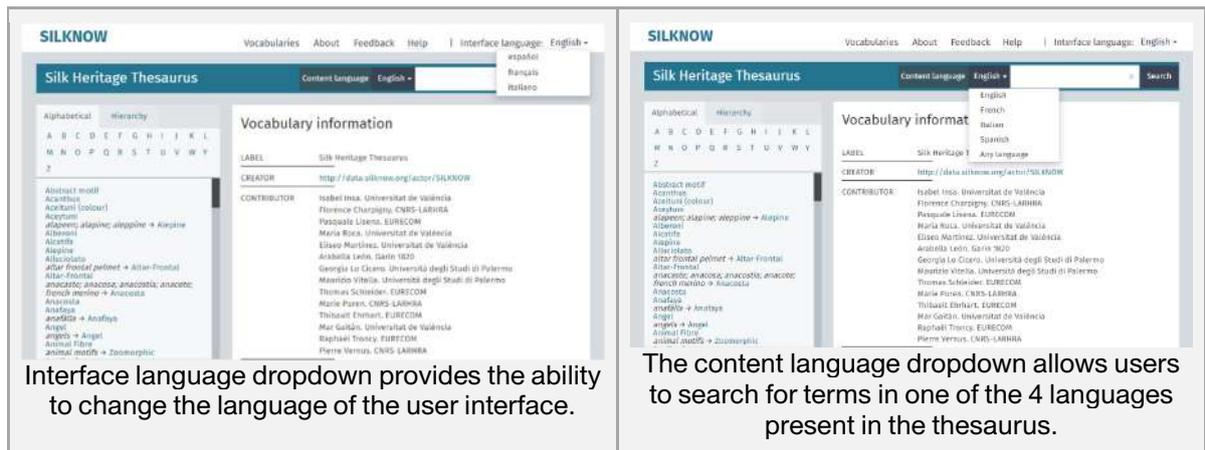


Figure 2. Screenshots of SILKNOW’s Thesaurus Browser to test functionality 02. The difference between changing/selecting the interface language (left) and the content language (right) is shown.

2.2.3. Functionality 03

Description of the functionality: Using a simple search interface.

Fulfilled: Yes.

Implementation/use: The search interface consists of one search box with one selector for the language of the terms. The search box includes autocomplete functionality to provide users matching terms as words are being written. Examples are shown in Figure 3, where a user is searching for a term in the search box.

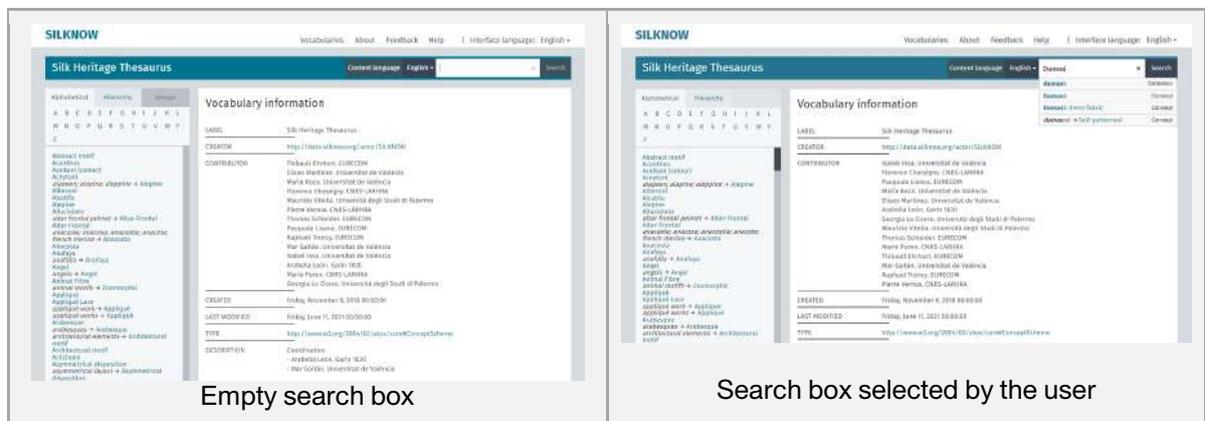


Figure 3. Screenshots of SILKNOW’s Thesaurus to test functionality 03.

2.2.4. Functionality 04

Description of the functionality: For queries that provide no answers (or for all queries), suggest related search terms in the same or other languages.

Fulfilled: Yes.

Implementation/use: This functionality is implemented using autocomplete in the search box. As depicted in Figure 4, users can start writing a term and the system will provide different options.

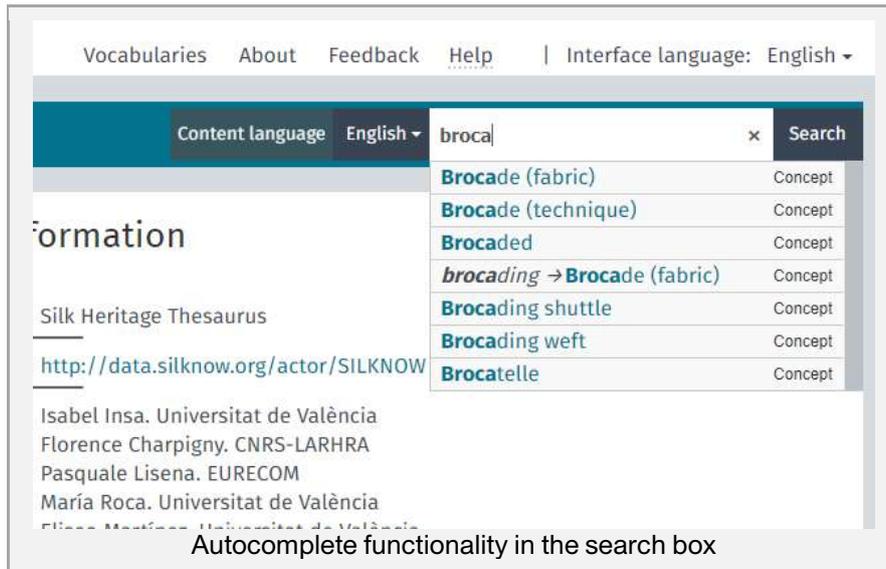


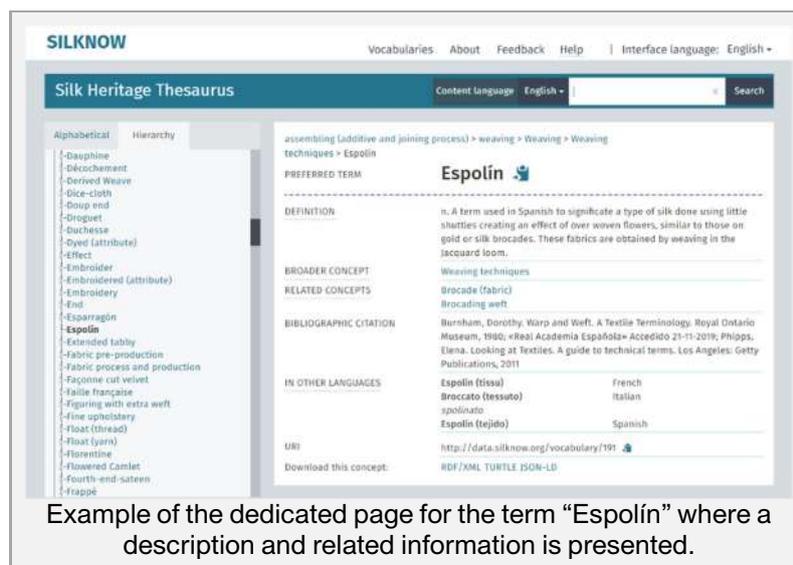
Figure 4. Screenshots of SILKNOW's Thesaurus to test functionality 04.

2.2.5. Functionality 05

Description of the functionality: Searching the thesaurus for terms and accessing their definition.

Fulfilled: Yes.

Implementation/use: Users have three different ways to search for terms: Alphabetically, Using Hierarchy dropdown, or using the Search box. Examples of searching alphabetically were shown in the previous examples, as part of Figures 1, 2 and 3. Examples of using a search box were shown in Figures 3 and 4. An example of a search using the hierarchy dropdown is given in Figure 5. In all these cases, clicking on the term opens up a dedicated page with the description and other related information.



Example of the dedicated page for the term “Espolín” where a description and related information is presented.

Figure 5. Screenshots of SILKNOW's Thesaurus to test functionality 04, making use of the Hierarchy to search options for the specific term “Espolín”.

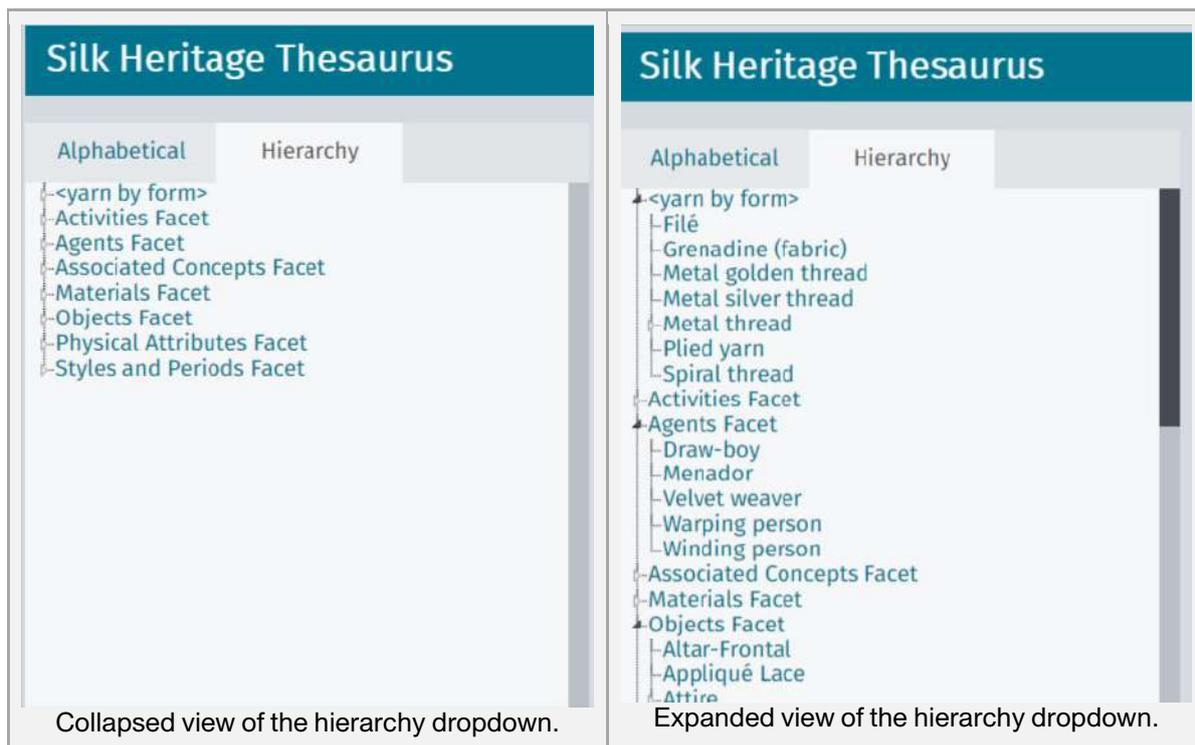


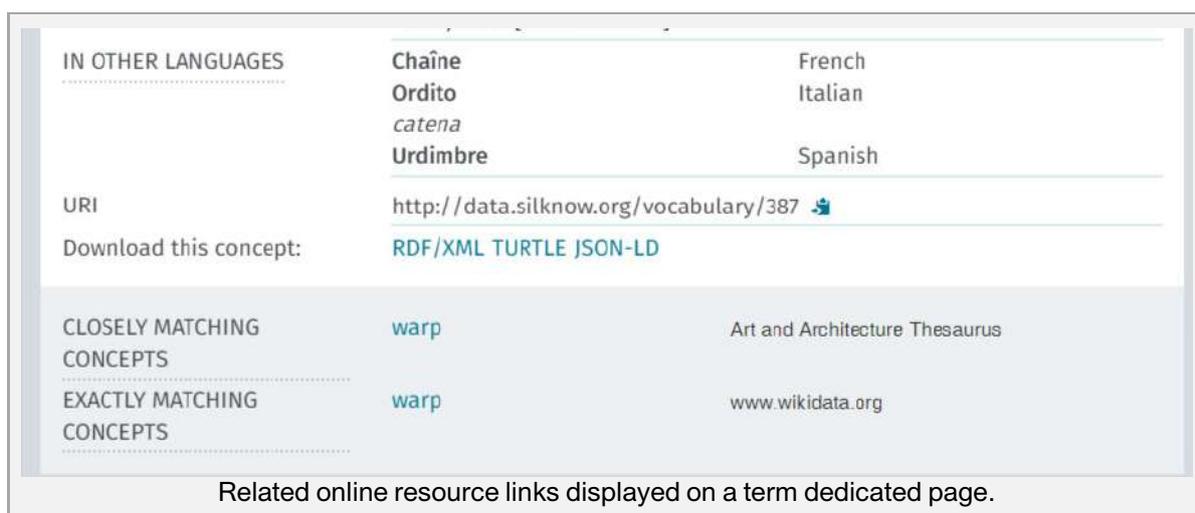
Figure 7. Screenshots of SILKNOW's Thesaurus to test functionality 07.

2.2.8. Functionality 08

Description of the functionality: Linking SILKNOW data to other online resources such as Europeana, Wikidata, IdRef, RAMEAU, Library of Congress authorities, etc.

Fulfilled: Yes.

Implementation/use: Each term page has a section for matching concepts provided by other online resources such as Wikidata or AAT. Links are provided in order to explore the term in those online resources. This is illustrated in Figure 8.



Related online resource links displayed on a term dedicated page.

Figure 8. Screenshots of SILKNOW's Thesaurus to test functionality 08.

2.2.9. Functionality 09

Description of the functionality: Retrieving bibliographic references related to a record, when available.

Fulfilled: Yes.

Implementation/use: For those records where bibliographic references are available, the references are displayed on the term page alongside the rest of related information. An example for the term “Warp (yarn)” is shown in Figure 9. As can be seen, this example involves different bibliographic references.

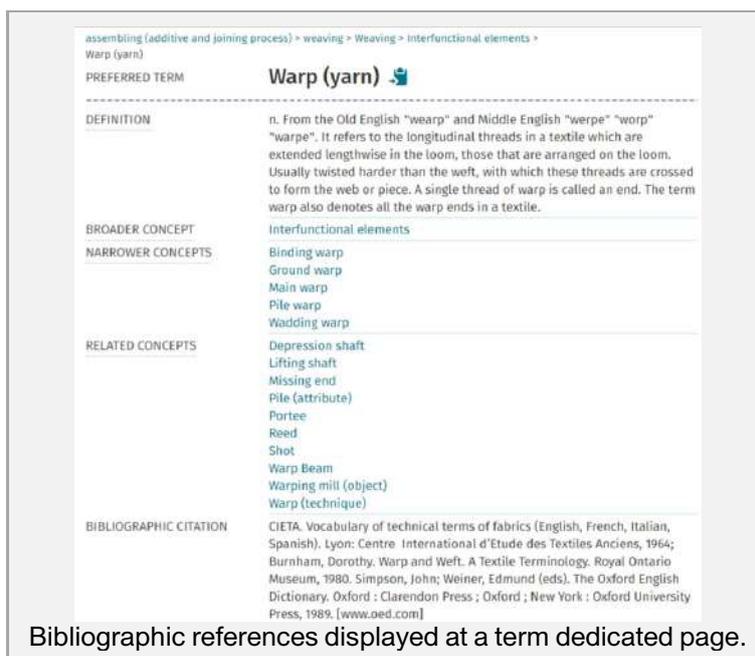


Figure 9. Screenshots of SILKNOW’s Thesaurus to test functionality 09.

2.2.10. Discussion

Functional testing for SILKNOW’s Thesaurus shows very satisfactory results for all the functionalities that were identified by SSH partners in the scope of WP2. All these functionalities have been integrated and can be accessed via the public web application enabling browsing of the SILKNOW Thesaurus.

2.3. Stress testing

With the stress testing of SILKNOW’s Thesaurus Browser, we aim to determine its robustness in producing the appropriate answer for the user. SILKNOW’s thesaurus browser offers a very usable user interface to analyse vocabulary content. To this end, we have prepared some experiments which consist of performing some interface actions using the four different languages (English, Spanish, French and Italian) and increasing the number of concurrent users. Our goal is to establish the number of concurrent users the current configuration can handle before performing horizontal scale-up of the infrastructure. The experiments and the hardware used are described in section 2.3.1. During the experiments we measured if the server response is adequate. The way to measure if a response is adequate is explained in

2.3.2. The data gathered during the testing is provided in section 2.3.3. An individual analysis per type of test performed is presented in sections 2.3.4, 2.3.5 and 2.3.6. Finally, a general analysis is discussed in section 2.3.7.

2.3.1. Test description

In order to perform the stress tests with the different APIs of the SILKNOW Thesaurus Browser, we prepared an experiment to be completed by different users. To this end, we defined a thread group made up of different numbers of users, namely 5, 10, 30 and 50 users (Table 2). These users are simulated by the computer, i.e., they are not real users. Once the users are defined, each of them launches a request in the four languages (English, Spanish, French and Italian). In this way, we intend to measure the response of the tool regarding three basic tasks:

- **Searching:** A request searching the term ‘damask’ in English, French, Spanish and Italian. This test emulates the process performed by a user when searching for a term in the thesaurus using the web user interface. This process is evaluated in the 4 languages: Searching “damasco” in Spanish activated “damask” in English, “damasco” in Italian and “damas” in French. The results of this test are shown and analysed in subsection 2.3.4.
- **Alphabetical:** This test emulates the process performed by a user when he/she clicks on a concept in the alphabetical list (also damask) in order to obtain all the information about a concept. Similarly, the request is repeated in English, French, Spanish and Italian. The test performs an action of the type HTTP GET on a concept. The results of this test are shown and analysed in subsection 2.3.5.
- **Hierarchy:** A request in English, French Spanish and Italian by clicking on the same concept damask, but on the hierarchy. This test emulates the process followed by a user when he/she clicks on a branch of the hierarchy in order to know what the narrower and broader concepts of the ‘damask’ focused concept are. The test performs a query, retrieving the narrower concepts to a general concept. The results of this test are shown and analysed in subsection 2.3.6.

During the tests, all the thread group users execute their requests concurrently. Finally, when the first iteration ends, it is repeated four times, i.e., each test is performed five times.

The same tests are executed using three different thesaurus access methods:

- SILKNOW’s Thesaurus Web Interface (SKOSMOS URI).
- SILKNOW’s Thesaurus SILNOW Public API.
- SILKNOW’s Thesaurus SILKNOW SPARQL API.

In order to obtain more information about the different SILKNOW Thesaurus APIs, deliverables D6.3 and D6.4 can be consulted, where these APIs are widely described. The structure of the thread groups and the requests performed are shown in Table 2. We also show the hour and date when the tests were initiated.

Number of users	Requests per user	Times repeated	Total requests	Time and date
SILKNOW THESAURUS - SKOSMOS URI				
5	Search: 4	5	100	2021-06-24T04:00:00+0000
	Hierarchy: 4		100	2021-06-24T04:30:00+0000

	Alphabetical: 4		100	2021-06-24T05:00:00+0000
10	Search: 4 Hierarchy: 4 Alphabetical: 4	5	200 200 200	2021-06-24T06:30:00+0000 2021-06-24T07:00:00+0000 2021-06-24T08:00:00+0000
30	Search: 4 Hierarchy: 4 Alphabetical: 4	5	600 600 600	2021-06-24T12:00:00+0000 2021-06-24T14:00:00+0000 2021-06-24T16:00:00+0000
50	Search: 4 Hierarchy: 4 Alphabetical: 4	5	1000 1000 1000	2021-06-24T18:00:00+0000 2021-06-24T20:00:00+0000 2021-06-24T22:00:00+0000
SILKNOW THESAURUS – SILKNOW PUBLIC API				
5	Search: 4 Hierarchy: 4 Alphabetical: 4	5	100 100 100	2021-07-01T15:00:00+0000 2021-07-01T15:30:00+0000 2021-07-01T16:00:00+0000
10	Search: 4 Hierarchy: 4 Alphabetical: 4	5	200 200 200	2021-07-01T17:00:00+0000 2021-07-01T17:45:00+0000 2021-07-01T18:30:00+0000
30	Search: 4 Hierarchy: 4 Alphabetical: 4	5	600 600 600	2021-07-01T19:30:00+0000 2021-07-01T20:30:00+0000 2021-07-01T21:30:00+0000
50	Search: 4 Hierarchy: 4 Alphabetical: 4	5	1000 1000 1000	2021-07-02T01:00:00+0000 2021-07-02T02:30:00+0000 2021-07-02T04:00:00+0000
SILKNOW THESAURUS – SILKNOW SPARQL API				
5	Search: 4 Hierarchy: 4 Alphabetical: 4	5	100 100 100	2021-07-02T05:00:00+0000 2021-07-02T05:30:00+0000 2021-07-02T06:00:00+0000
10	Search: 4 Hierarchy: 4 Alphabetical: 4	5	200 200 200	2021-07-02T06:30:00+0000 2021-07-02T07:30:00+0000 2021-07-02T08:30:00+0000
30	Search: 4 Hierarchy: 4 Alphabetical: 4	5	600 600 600	2021-07-02T10:00:00+0000 2021-07-02T11:30:00+0000 2021-07-02T13:00:00+0000
50	Search: 4 Hierarchy: 4 Alphabetical: 4	5	1000 1000 1000	2021-07-02T14:00:00+0000 2021-07-02T16:00:00+0000 2021-07-02T18:00:00+0000

Table 2. List of the different tests, requests and concurrent users per test in the different SILKNOW's Thesaurus APIs (SKOSMOS URI, SILKNOW Public, SPARQL).

Additionally, the tests were designed based on the following considerations:

- **Sequence.** The requests are sequentially performed per user. There is a random timer, from 0 to 3 seconds between every request. Due to this, the second request could start one second, two seconds, or three seconds after the previous request. So, the maximum life of the user's thread is 12 seconds for 4 requests, plus the response time of the performed requests.
- **Delay.** The ramp-period value, which defines the delay to the next user thread starting, is defined in each test to wait 3 seconds before the next user performs the first request. So, when a user launches the first request, the first request of the next user is performed three seconds later.

To clarify the process, Figure 10 depicts a test execution sample for users I, II and III. As can be seen, user I launches three requests. The time required to execute each request is the elapsed time associated with the request, and a random timer (0-3 seconds) per request. So, user II starts the first request after user I starts his/her first query and passes the ramp-up

SILKNOW

period (3 seconds). Finally, user III starts the first query after user II starts his/her first query and passes the ramp-up period (3 seconds from user II, and 6 seconds from user I).

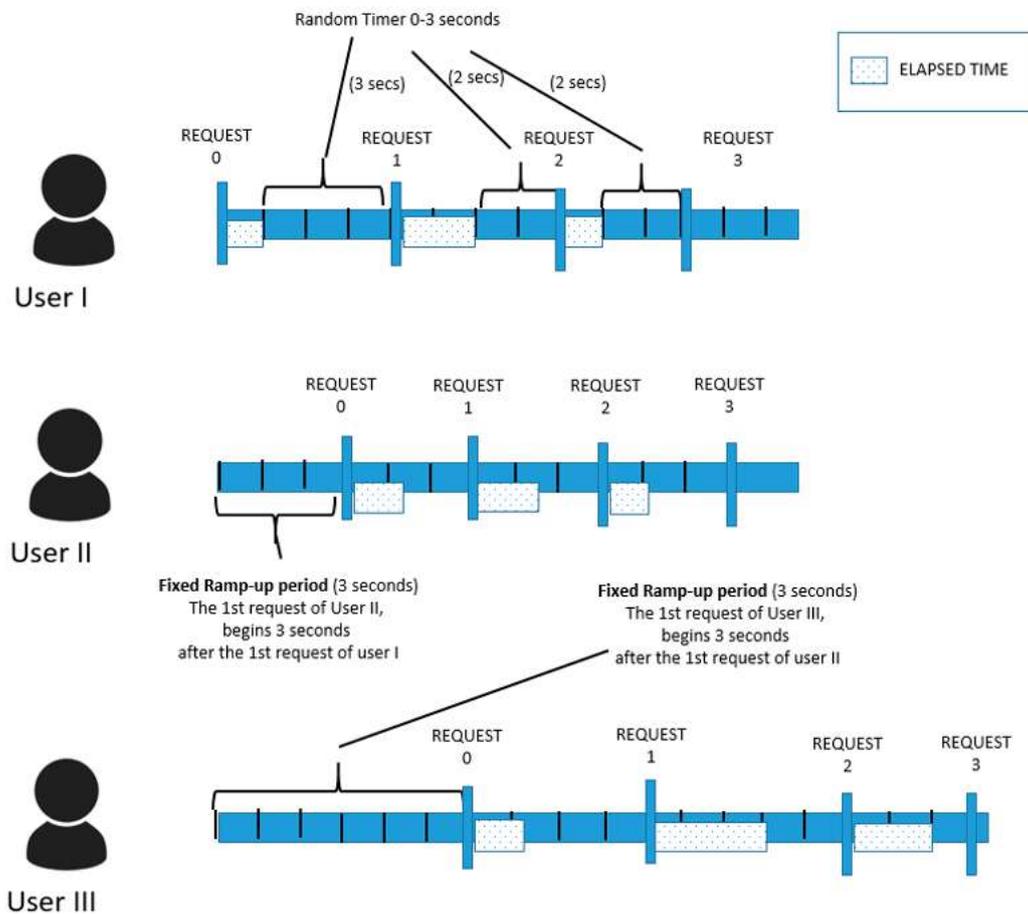


Figure 10. A request execution sample for users I, II and III in a thread group of 5 users.

The hardware configuration that supports the APIs services is described as follows:

- CPU: Intel Xeon L5640, 2.26 GHz, 12 cores (24 threads).
- RAM: 128 GB.
- Operating System: Linux Debian Buster, kernel 4.12.0.

The Knowledge Graph is hosted in a Virtuoso Docker replicated in a twin component. A load balancer between the two images is used to distribute the server load.

The client system where the tests were executed has the following characteristics:

- CPU: Intel i5-6400 CPU @ 2.70GHz.
- RAM: 8GB.
- Operative System: Linux Fedora 7.0.

The tests were launched from the JMeter tool [4] using OpenJDK Java 1.8. We used this tool because of the API features and the test requirements which make Apache JMeter an appropriate tool to perform such tests [5].

2.3.2. How to evaluate the result of a request

In this section, we explain how we measure and define if a response is adequate. In order to evaluate the response generated by the server, the JMeter tool allows the definition of one, or many, Response Assertions. Specifically, in the thesaurus test there were two kinds of response assertion defined per request:

- **Response 1.** The first response checks that there were no basic HTTP Errors, or Exceptions. It checks that response headers contain the string “HTTP/1.1 200 OK”. When an unexpected error page, or exception page, is generated from the server this content is not present in the response headers.
- **Response 2.** The second one checks that the response text contains terms that appear in the normal response. For instance, searching for the term “damask” in English the response assertion checks that the following terms are present in the response text:
 - Damask
 - Damask dress fabric.

The first response assertion is the same for all the requests performed in the test, while the second response assertion is also present in all the tests, but the terms change according to the request. Table 3 enumerates the tests and requests, and the expected response text for each of them. Therefore, if we obtain a different term from the ones expected, we assume that the result of a request is wrong.

Test	Request	Expected terms in response text
Search	English, Damask	Damask, Damask dress fabric
	Spanish, Damasco	Damasco, Damasco de vestido
	French, Damas	Damas, Damas robe
	Italian, Damasco	Damasco, Damasco bicolore
Alphabetical	Get the concept Acanthus (English)	Acanthus, Vegetal Motif
	Get the concept Acanto (Spanish)	Acanto Motivo vegetal
	Get the concept Acanthe (French)	Acanthe Motif végétal
	Get the concept Acanto (Italian)	Acanto Motivo vegatale
Hierarchy	Get narrower terms in English to the concept with URI http://vocab.getty.edu/aat/300264090	Aceytuni, Batik
	Get narrower terms in Spanish to the concept with URI http://vocab.getty.edu/aat/300264090	Aceituní, Batik
	Get narrower terms in French to the concept with URI http://vocab.getty.edu/aat/300264090	Aceituni, Batik
	Get narrower terms in Italian to the concept with URI http://vocab.getty.edu/aat/300264090	Aceituni, Bathik

Table 3. Expected terms in the response text, per request, in order to be accepted.

2.3.3. Gathered data during the tests

As introduced above, in this section we show the data gathered during the tests which will then be analysed in subsections 2.3.4, 2.3.5 and 2.3.6. Tables 4, 5 and 6 give a summary of the results of the tests, where the values presented are the average of the elapsed time per request and type of test, and the percentage of fails per request and type of test. Regarding the “fails percentage”, we measure if the response text is exactly the one that we expected, as explained in section 2.3.2 and summarized in Table 3. Therefore, a result of “0” for this field means that all the response texts for a certain test are correct.

	CONCURRENT USERS	DATA	REQUEST				
			ENGLISH	SPANISH	FRENCH	ITALIAN	AVERAGE
SEARCH	5	Average Elapsed time	743	969	2523	966	961
		Fails percentage	0	0	0	0	0
	10	Average Elapsed time	916	1005	2523	1011	1361
		Fails percentage	0	0	0	0	0
	30	Average Elapsed time	954	1128	3443	1144	1667
		Fails percentage	0	1.3	0	0	0.3
	50	Average Elapsed time	1240	1030	4500	1305	2019
		Fails percentage	0	20.4	11.6	8.4	10.1
ALPHABETICAL	5	Average Elapsed time	4304	4463	4017	4619	4238
		Fails percentage	0	0	0	0	0
	10	Average Elapsed time	5776	5112	4467	5248	5151
		Fails percentage	0	0.4	0	0	0.1
	30	Average Elapsed time	13337	11613	9793	11028	11443
		Fails percentage	1.5	3.5	2	3	2.5
	50	Average Elapsed time	21479	19352	17616	17223	18917
		Fails percentage	29.6	42	2.8	5,6	20
HIERARCHY	5	Average Elapsed time	12976	12871	12969	12838	12913
		Fails percentage	0	0	0	0	0
	10	Average Elapsed time	13604	13551	13227	13202	13396
		Fails percentage	0	0	0.2	0.2	0.1
	30	Average Elapsed time	19326	19550	19582	19902	19387
		Fails percentage	1.3	0.6	0.6	1,3	1
	50	Average Elapsed time	30311	30225	30325	30348	30302
		Fails percentage	97	97	99	100	98

Table 4. Results of the test performed with SILKNOW’s Thesaurus Browser with the SKOSMOS URI. The table contains the average of the elapsed time per request and type of test, as well as the percentage of fails per request and type of test.

	CONCURRENT USERS	DATA	REQUEST				
			ENGLISH	SPANISH	FRENCH	ITALIAN	AVERAGE
SEARCH	5	Average Elapsed time	1118	786	879	850	908
		Fails percentage	0	3	4	8	5
	10	Average Elapsed time	2225	2207	877	988	1574
		Fails percentage	0	0	4	0	1
	30	Average Elapsed time	1329	768	774	748	905
		Fails percentage	0	3	1	0	1
50	Average Elapsed time	1720	725	732	744	980	
	Fails percentage	0	2.4	1.6	1.2	1.3	
ALPHABETICAL	5	Average Elapsed time	1084	796	751	812	861
		Fails percentage	0	0	0	0	0
	10	Average Elapsed time	1063	792	709	770	833
		Fails percentage	0	0	2	0	0.5
	30	Average Elapsed time	1285	778	730	742	884
		Fails percentage	0	0	2.6	0.6	0.8
50	Average Elapsed time	1560	738	727	728	938	
	Fails percentage	0	0	0.4	1.6	0.5	
HIERARCHY	5	Average Elapsed time	1106	831	879	759	894
		Fails percentage	0	0	0	4	1
	10	Average Elapsed time	1139	781	753	753	857
		Fails percentage	0	0	2	2	1
	30	Average Elapsed time	1524	801	750	747	965
		Fails percentage	0	0	2	0,5	0.5
50	Average Elapsed time	1506	797	837	768	977	
	Fails percentage	0	1.2	1.6	2	1.2	

Table 5. Results of the test performed with SILKNOW's Thesaurus Browser with the SILKNOW Public API. The table contains the average of the elapsed time per request and type of test, as well as the percentage of fails per request and type of test.

	CONCURRENT USERS	DATA	REQUEST				
			ENGLISH	SPANISH	FRENCH	ITALIAN	AVERAGE
SEARCH	5	Average Elapsed time	387	128	144	137	199
		Fails percentage	0	0	0	0	0
	10	Average Elapsed time	507	135	134	114	223
		Fails percentage	0	4	2	0	1,5
	30	Average Elapsed time	434	122	165	142	216
		Fails percentage	0	0	0.6	0.6	0.3
50	Average Elapsed time	448	179	123	141	223	
	Fails percentage	0	1.2	1.2	1.6	1	
ALPHABETICAL	5	Average Elapsed time	1050	766	770	745	833
		Fails percentage	0	0	0	0	0
	10	Average Elapsed time	1063	766	726	713	817
		Fails percentage	0	0	4	4	2
	30	Average Elapsed time	1330	728	753	738	887
		Fails percentage	0	0	0.6	1.3	0.5
50	Average Elapsed time	1491	754	743	757	936	
	Fails percentage	0	2.8	1.2	2	1.5	
HIERARCHY	5	Average Elapsed time	419	154	158	135	217
		Fails percentage	0	0	0	0	0
	10	Average Elapsed time	378	158	178	187	225
		Fails percentage	0	2	0	0	0.5
	30	Average Elapsed time	420	171	134	158	221
		Fails percentage	0	0.6	0.6	0	0.3
50	Average Elapsed time	481	134	165	176	239	
	Fails percentage	0	0.8	2	2.4	1.3	

Table 6. Results of the test performed with SILKNOW's Thesaurus Browser with the SILKNOW SPARQL API. The table contains the average of the elapsed time per request and type of test, as well as the percentage of fails per request and type of test.

2.3.4. SKOSMOS URI

2.3.4.1. Results and analysis for SEARCH tests SKOSMOS URI

Figure 11 shows a graphic with the mean of the elapsed time and the fails percentage per request in the test with 5, 10, 30, and 50 concurrent users, elaborated with the data presented in Table 4 for the SEARCH tests in the SKOSMOS URI. Overall, we can assess that the stress tests related to the search term process in the thesaurus obtain good results in the average of the elapsed time per request. Overall, we observe that the elapsed time increases as the number of users increases, which is normal because the server is working with more requests. For Spanish, Italian and English, the mean value of this time is very close to one second, even with 50 concurrent users. On the other hand, greater values are obtained for French, as the mean time is up to 4 times longer (with 50 concurrent users) than for the other

languages. This fact can be related to the execution order of the queries. Retrospectively, we should probably have shuffled the order of languages between each test to neutralize the caching effect.

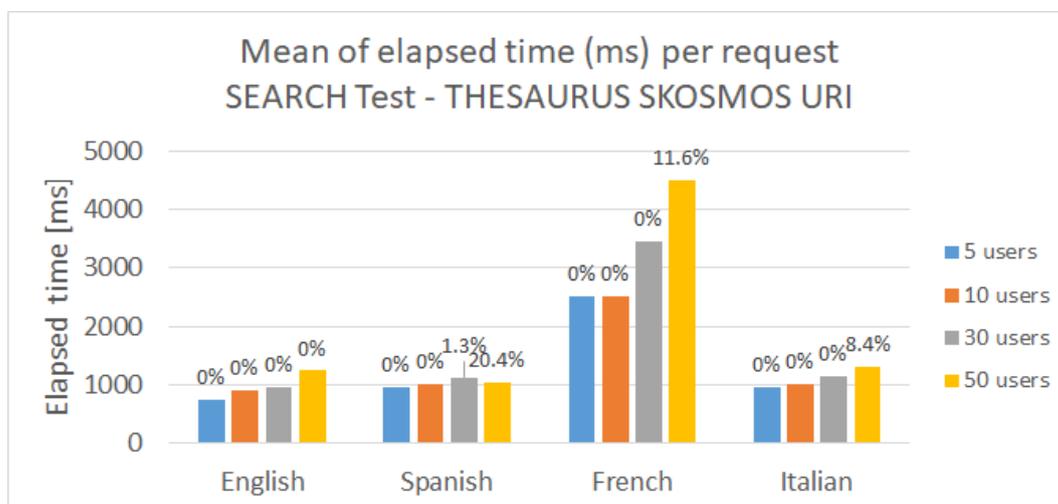


Figure 11. Mean of the elapsed time per request searching for a term in different languages with tests with 5,10,30 and 50 concurrent users with the SKOSMOS URI. The numerical values depicted on each column represent the fails percentage for the corresponding request.

The percentage of fails in this test is very low. The maximum value is reached for the test with 50 concurrent users for Spanish (20.4%). As an average, this value is 10.1% for 50 concurrent users, and 0.2% with 30 concurrent users, while the rest of the tests had no fails.

The recommended number of concurrent users executing this task is 30, given the current server configuration.

2.3.4.2. Results and analysis for ALPHABETICAL tests (Get concept) with SKOSMOS URI

Figure 12 shows a graphic with the mean of the elapsed time and fails percentage per request in the test with 5, 10, 30, and 50 concurrent users, elaborated with the data presented in Table 4 for the ALPHABETICAL tests in the SKOSMOS URI. As can be seen, the average of the elapsed time per request with 5 and 10 concurrent user tests was close to 5 seconds per request, but it was a little bit longer with 30 concurrent users, as it took 10 seconds per request, and worse with 50 users, taking more than 15 seconds per request. As expected, the elapsed time increased with an increasing number of users. Overall, we can say that the ALPHABETICAL stress tests, which requests getting the information about a concept, obtains reasonable results through the SKOSMOS URI for 5 and 10 users, but with 30 and 50 concurrent users the tool's performance is not adequate for a good user experience given the current server configuration.

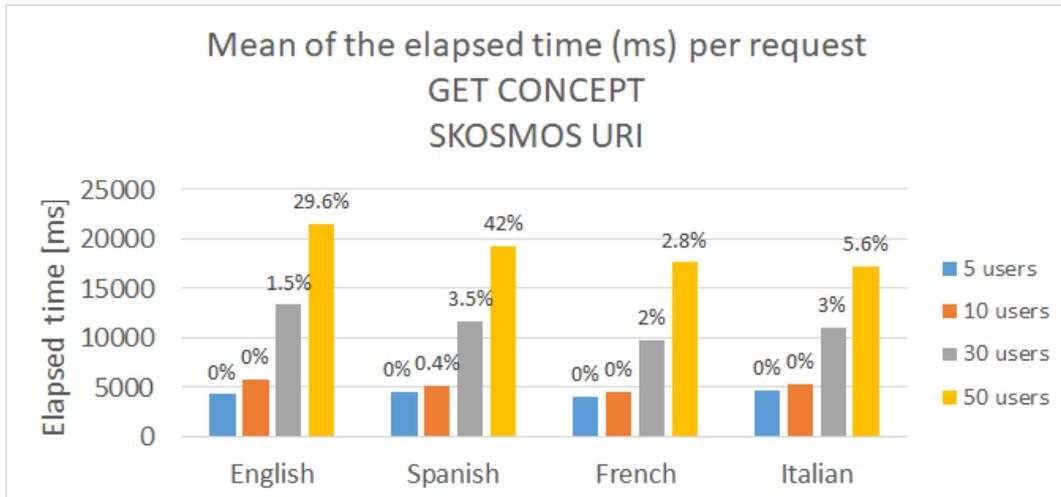


Figure 12. Mean of the elapsed time per request Get CONCEPT (alphabetical) list in different languages with 5, 10, 30 and 50 concurrent users with the SKOSMOS URI. The numerical values depicted on each column represent the fails percentage for the corresponding request.

The percentage of fails is also greater in the tests with 50 concurrent users, reaching an average of 20%. In the tests with 30 concurrent users, the average percentage of fails is 2.5%. For the rest of the tests, the average percentage of fails remains low, mostly 0%.

The recommended concurrent users executing this task is 10, given the current server configuration.

2.3.4.3. Results and analysis for HIERARCHY tests (query for getting the narrower concepts) with SKOSMOS URI

Figure 13 shows a graphic with the mean of the elapsed time and the fails percentage per request in the test with 5, 10, 30 and 50 concurrent users, elaborated with the data presented in Table 4 for the HIERARCHY tests in the SKOSMOS URI.

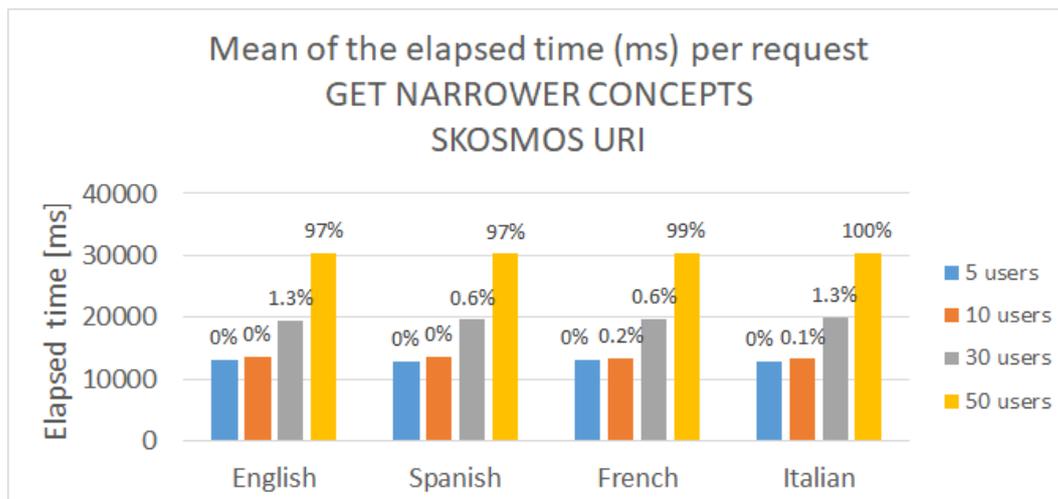


Figure 13. Mean of the elapsed time per request getting the narrower concepts with 5, 10, 30 and 50 concurrent users with the SKOSMOS URI. The numerical values depicted on each column represent the fails percentage for the corresponding request.

Overall, the stress tests related to getting the narrower term to a concept in the thesaurus achieved poor results when using the SKOSMOS URI. As can be seen, the elapsed time increases as the number of users increases. With 5 and 10 concurrent users, the mean of the elapsed time is greater than 12 second per request, which is a higher value than the ones obtained with the other APIs. With 30 concurrent users, this time is close to 20 seconds per request, and 30 seconds with 50 concurrent users.

Regarding the values obtained for the percentage of fails, they were very good for tests with 5, 10 and 30 concurrent users, as the average reaches a maximum of 1% for 30 users. However, with 50 concurrent users, the tool cannot manage the request frequency, as the average percentage of failures reaches 98%, probably due to the timeout of the SKOSMOS internal API.

To sum up, the current server configuration can manage up to 30 concurrent users with an acceptable number of failures. However, the response time is long, close to 20 seconds. This time is not acceptable for a good user experience. Therefore, we recommend 10 users as the maximum number of concurrent users in this case, and we note that this access method should be optimized to improve user experience.

2.3.5. SILKNOW PUBLIC API

2.3.5.1. Results and analysis for SEARCH tests with SILKNOW public API

Figure 14 shows a graphic with the mean of the elapsed time and the fails percentage per request in the test with 5, 10, 30, and 50 concurrent users, built from the data presented in Table 5 for the SEARCH tests using the SILKNOW public API. As can be seen, all the obtained values for the elapsed time are lower than 2.5 seconds, and therefore in all cases they are acceptable for a good user experience.

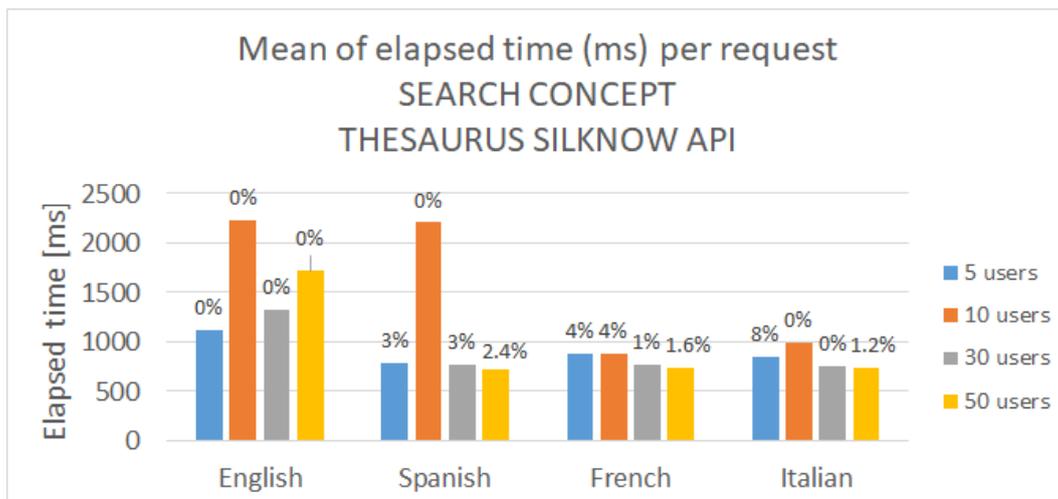


Figure 14. Mean of the elapsed time per request searching for a term in different languages with tests with 5, 10, 30 and 50 concurrent users with the SILKNOW Public API. The numerical values depicted on each column represent the fails percentage for the corresponding request.

Overall, the stress tests related to the search term process in the thesaurus achieved good results for the average elapsed time per request. The tests performed on Spanish, Italian and French have an average time value of very close to one second. Only the tests with 10 concurrent users have a value a bit longer than 2 seconds for Spanish. If this exceptional result in the test with 10 concurrent users in Spanish is not taken into account, since it could be due to external traffic or network conditions, we can assume that the number of users is not very important for the results of the tests, at least up to 50 concurrent users. All the tests executed with requests in English have values between 1 and 2 seconds. These requests are the first ones executed, and the caching system makes the rest of them require less time.

The retrieved values of the percentage of fails are also reasonable. 45% of the tests have no fails, and the rest of the values are lower than 4%, with the only exception being the test with the Italian query for 5 concurrent users, which has a fail percentage value of 8%.

The recommended number of concurrent users executing this task is 50, with a similar hardware configuration to the one used in these tests.

2.3.5.2. Results and analysis for ALPHABETICAL tests (Get concept) with SILKNOW public API

Figure 15 shows a graphic with the mean elapsed time and the fails percentage per request in the test with 5, 10, 30, and 50 concurrent users, prepared with the data presented in Table 5 and for the ALPHABETICAL tests using the SILKNOW public API. As can be seen, all the gathered data had low values. As an average, the results with English requests are a bit higher than the results of the tests with the other languages. This is due to the caching system of the API, but the values are still under 1.6 seconds. Therefore, it is safe to say that for all cases the values retrieved from the tests are adequate for a good user experience.

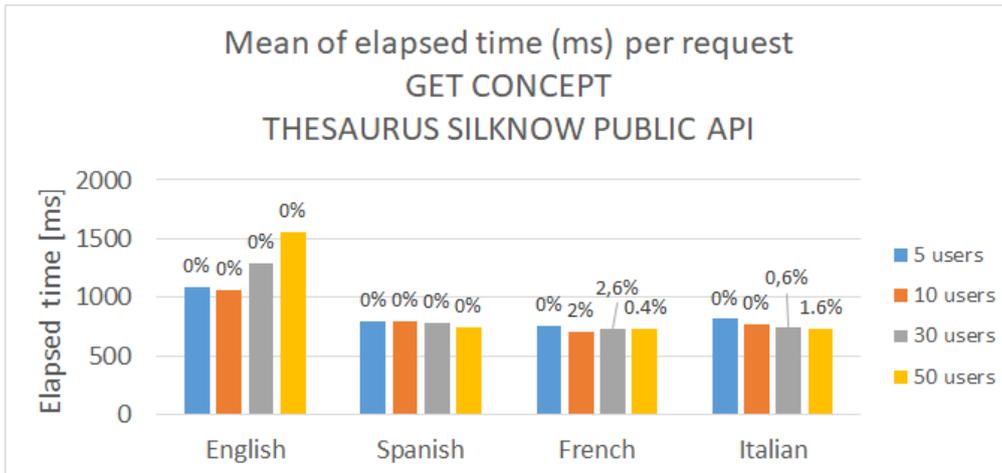


Figure 15. Mean of the elapsed time per request searching for a term in different languages with tests with 5, 10, 30 and 50 concurrent users with the SILKNOW Public API. The numerical values depicted on each column represent the fails percentage for the corresponding request.

The percentages of fails in this test are very low. Most of the requests have no failures; only in 5 of the requests do we find values with some fails, with a maximum of 2.6%.

2.3.5.3. Results and analysis for HIERARCHY tests (query for getting the narrower concepts) with SILKNOW Public API

Figure 16 shows a graphic with the mean of the elapsed time and the fails percentage per request in the test with 5, 10, 30, and 50 concurrent users, elaborated with the data presented in Table 5 and for the HIERARCHY tests using the SILKNOW public API. As in the previous set of tests, the delays with the English requests are a bit longer due to the caching system of the API. However, as can be seen, all obtained data have elapsed time values below 1.6 seconds, even with 50 concurrent users, which is a low value. Therefore, results are satisfactory for a good user experience, in all cases.

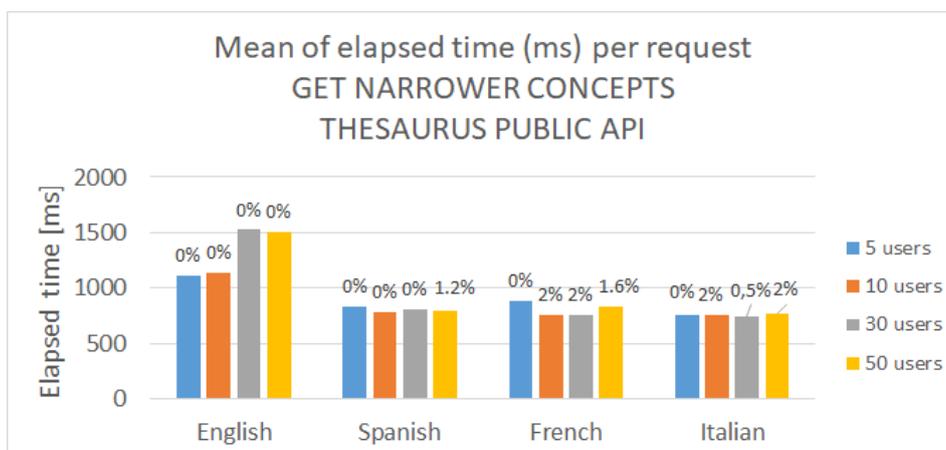


Figure 16. Mean of the elapsed time per request searching for a term in different languages with tests with 5, 10, 30 and 50 concurrent users with the SILKNOW Public API. The numerical values depicted on each column represent the fails percentage for the corresponding request.

The percentage of fails in these tests is also very low. Only 7 of the tests show fail values, while the maximum fails percentage value is 2%. The rest of tests had no fails.

The recommended number of concurrent users executing this task is, again, 50 with a similar hardware configuration to that used in these tests.

2.3.6. SILKNOW SPARQL API

2.3.6.1. Results and analysis for SEARCH tests with SILKNOW SPARQL API

Figure 17 shows a graphic with the mean of the elapsed time and the fails percentage per request in the test with 5, 10, 30, and 50 concurrent users, prepared with the data presented in Table 6 using the plain SPARQL API. The elapsed time increased as the number of users increased, but still with very low values (with a maximum of 507ms). As with the previous tests, the English values required a longer time than other languages due to the caching system. As all the retrieved values are in the margin of 0.5 seconds, or lower, all of them provide a good user experience.

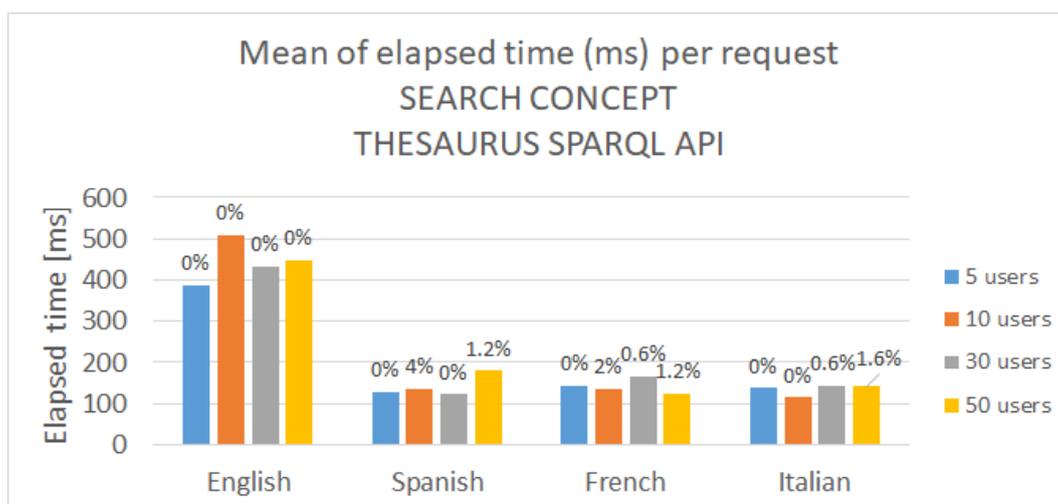


Figure 17. Mean of the elapsed time per request searching for a term in different languages with tests with 5, 10, 30 and 50 concurrent users with the SILKNOW SPARQL API. The numerical values depicted on each column represent the fails percentage for the corresponding request.

The percentage of fails in this test were also low. Only one test reached a failure percentage value of 4%. The remaining tests that showed failure values (a total of 6 tests) have values lower than 2%.

The recommended number of concurrent users executing this task can go up to 50 (and probably more) with the same hardware configuration as that used in these tests.

2.3.6.2. Results and analysis for “Alphabetical” tests (Get concept) with SILKNOW SPARQL API

Figure 18 shows a graphic with the mean of the elapsed time and the fails percentage per request in the test with 5, 10, 30, and 50 concurrent users, built from the data presented in Table 6 using the SPARQL API. As can be seen, the average elapsed time is always lower than 1.6 seconds, even with 50 concurrent users. So, with these elapsed times the users’ experience can be safely assumed to be very good.

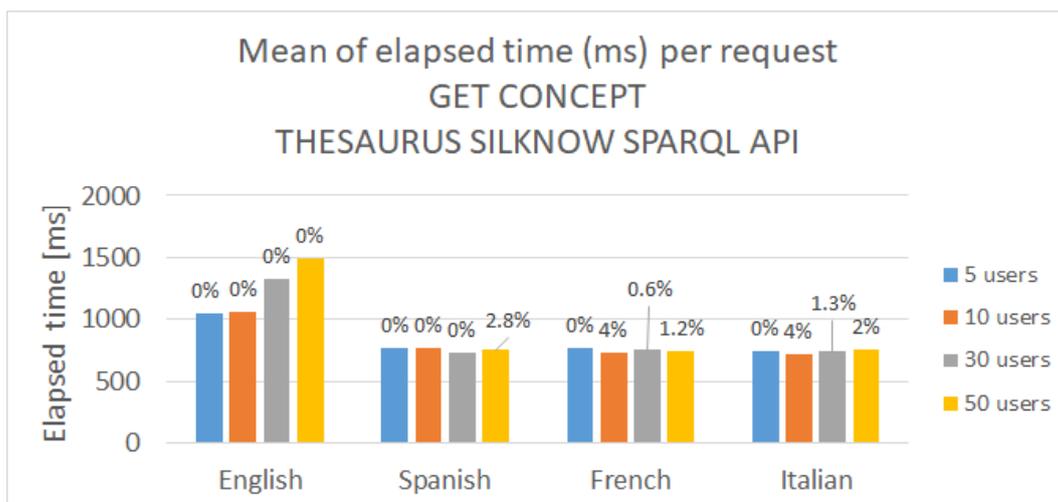


Figure 18. Mean of the elapsed time per request searching a term in different languages with tests with 5, 10, 30 and 50 concurrent users with the SILKNOW SPARQL API. The numerical values depicted on each column represent the fails percentage for the corresponding request.

The percentage of fails in this test is very low. Only in 7 of the tests are some values above 0%, but they are all lower than 4%. The remaining requests have no fails.

The recommended number of concurrent users executing this task is 50, with the same hardware configuration as the one used in these tests.

2.3.6.3. Results and analysis for “hierarchy” tests (query for getting the narrower concepts) with SILKNOW SPARQL API

Figure 19 shows a graphic with the mean of the elapsed time and the fails percentage per request in the test with 5, 10, 30, and 50 concurrent users, prepared with the data presented in Table 6 using the SPARQL API. In this case, the average elapsed time is always lower than 0.5 seconds, even with 50 concurrent users. Therefore, we can assure a very satisfactory user experience.

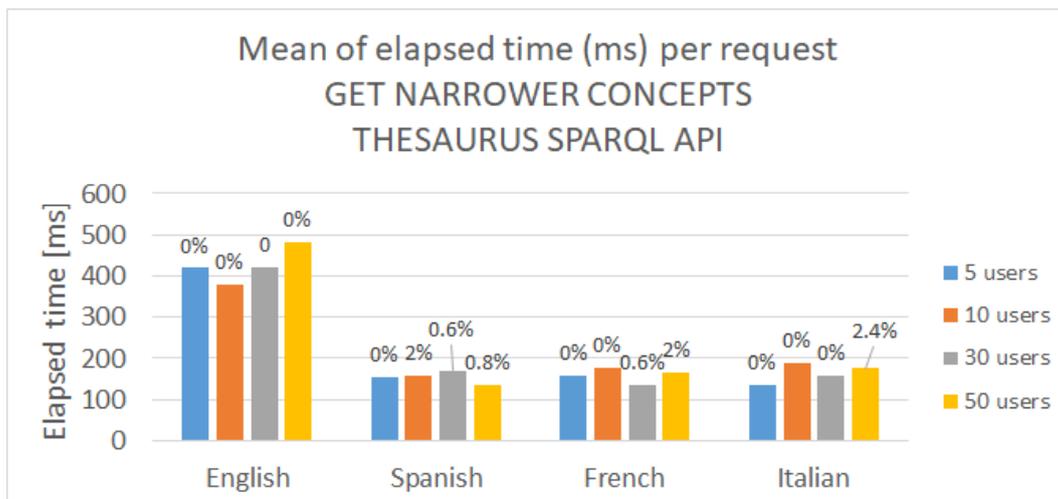


Figure 19. Mean of the elapsed time per request searching for a term in different languages with tests with 5, 10, 30 and 50 concurrent users with the SILKNOW SPARQL API. The numerical values depicted on each column represent the fails percentage for the corresponding request.

The percentage of fails in this test is very low. Only 6 tests failed, and the percentage values of fails are lower than 2.4%; the remaining requests have no fails.

In a similar way to the other tests performed for the SPARQL API, the recommended number of concurrent users executing this task is 50 with the same hardware configuration as that used in these tests.

2.3.7. Discussion

In the previous subsections (2.3.4.1 to 2.3.4.3), we analysed the results for the three tasks using SKOSMOS as the thesaurus access method. In this section, we further comment about the performance of using an API (this being the plain SPARQL API or the SILKNOW API) to access the thesaurus. While SKOSMOS is targeted at end users using a Web browser to search and browse the thesaurus, the API access is targeted at third party integrators looking to re-use the thesaurus.

The different tasks we have considered yield different results regarding the average elapsed time and the fails percentage. The best results are obtained for the SEARCH term tests that are optimized, regardless of the access method. Given the current hardware configuration, we can recommend up to 30 concurrent users using SKOSMOS and 50 concurrent users, or more, using an API. The two other tasks (getting details about a concept and getting narrower / broader concepts of a given concept) are more complex. While the two API access methods can handle these tasks well with up to 50 users, SKOSMOS performs worse. We recommend limiting the number of concurrent users to 5 using SKOSMOS for these two tasks, given the current hardware configuration. Improvements can naturally be obtained using horizontal scale-up of the infrastructure (i.e., adding more servers to the load balancer). We also recommend further optimizing the SKOSMOS software which can probably optimize these tasks better.

3. ADASILK

3.1. Description of ADASilk

ADASilk is a web-based search application that aims to provide access to the data stored in the Knowledge Graph (KG) through a public RESTful API. This API is developed and documented using:

- Grlc, a service for generating a web API from SPARQL queries which are versioned in a Git repository.
- SPARQL Transformer, a library that rewrites the output of SPARQL queries in a more suitable format for web development.

The user interface of the web application is developed using React, one of the most popular front-end Javascript libraries. The API is used to perform requests to the Knowledge Graph, the result is then rendered as HTML and sent to the users' browsers.

3.2. Evaluation of the functionalities identified by SSH experts

As given in the introduction section, the functionalities that should be offered by ADASilk were identified by SSH partners in the scope of WP2. From the table that they elaborated with the description of functionalities, in Table 7 we present those related to ADASilk. As can be seen, a total of 26 functionalities are identified. Similar to SILKNOW's Thesaurus Browser (Table 1), the table includes examples and additional comments to better explain the functionalities that the tool should bring, as well as to what user profiles or "personas" and sectors this can be of relevance. To this table, we have added a last column, in grey, that summarizes the results of the functionality evaluation which are presented in the following subsections (3.2.1 to 3.2.26). This functional evaluation aims to show which functionalities are successfully integrated into ADASilk and how this integration is achieved. Finally, a discussion is given (in 3.2.27).

Functionality ID	Description of the functionality	Examples, additional comments	Personas	Sectors	Fulfilled?
<i>General functionality: Having access to a more personalized experience</i>					
01	Having a personal account		All	All	Yes
02	Being able to save favorite images		Design student	Research and Education	Yes
03	Being able to create and share a personal selection of items, kind of a curated exhibition				Yes
04	Having access to information about the last updates and the new materials	A list of new additions and changes to the repository	All	All	Not yet
05	Making all our web resources responsive to all		All	All	Yes

SILKNOW

	user devices (desktop/laptop, tablet, smartphone)				
06	Choosing the language		All	All	Yes
<i>General functionality: Searching</i>					
07	Using a simple search interface	Based on metadata (text- only)	All	All	Yes
08	Using an advanced search interface and filtering results with facets		All	All	Yes
09	Using an advanced search interface and filtering results with facets: historical period and/or geographical origin		Visitor Fan of fashion	Cultural heritage and leisure	Yes
			Student in textile history and technology	Research and Education	
			Fashion journalist		
10	Using an advanced search interface and filtering results with facets: weaving techniques and fabrics		Middle-aged museum visitor	Cultural heritage and leisure	Yes
			All	Research and Education	
11	Using an advanced search interface and filtering results with facets: motifs, decorative patterns		All	All	Yes
12	Make a query based on a single image, uploaded by the user, or one selected in the repository. Give the option to include metadata analysis (as all of the above) or to search only by visual similarity.	It would apply only to textiles with a limited range of colors or shades (damasks, for instance). One alternative example: https://www.europeana.eu/portal/fr/explore/colours.html	Designers and design students	Creative and textile industries	Yes
<i>General functionality: Finding and visualizing</i>					

SILKNOW

13	Sorting the results by field (alphabetical order for text fields, asc or desc order for numerical or date fields) and also with facets (such as “by country”)		All	All	Yes
14	Using a zoom tool for visualization		All	All	Yes
15	Provide the user with a link (when available) back to the online catalogue of the owning institution, to get more information, etc.	Only possible in cases when records already have permanent identifiers in the owning institution’s website.	All	All	Yes
16	Being able to find other SILKNOW related records by clicking on keywords or CHIs or authors... etc.	As in any OPAC or online catalog.	College student		Yes
17	Being able to find other SILKNOW records of related objects (clickable list of related records, if this information was included in the original records)	For instance, when a textile is part of a larger ensemble of more pieces, and the user wants to recover the entire ensemble.			Yes
18	Having access to the name of the owning institution in the record and being able to click on their names for more information (location, contacts...).		Museum curator		Not yet
19	Clicking on a weaving technique in order to access a detailed analysis of it in the Virtual Loom, when available		Fashion student Museum director	Research and Education	Yes
<i>General functionality: Downloading</i>					
20	Querying our database through an API, not just through a web interface		Museum curator	Research and Education	Yes
21	Downloading a list of selected results with a standard, basic set of metadata		Museum curator	Research and Education	Yes

22	Downloading individual search results (one record) in the format chosen by the user: image only in bitmap format (jpg, png, etc.), metadata only, entire record (pdf, csv, rtf...), etc.		Visitor, conservator	Cultural heritage and leisure	Yes
			Student	Research and Education	
<i>General functionality: Sharing</i>					
23	Being able to share content (one single record) on social networks (e.g. Facebook, Instagram, Pinterest...), by email, with a URL.		All	All	Yes
24	Being able to share multiple content: (a query, a user selection of records, etc) by the same procedures as above (social media, email, URL, etc.).		All	All	Yes
25	Providing the user with clear information about the licences / authors' rights applied on the images and what is possible to do or not with them.		All	All	Yes
<i>General functionality: Contacting</i>					
26	Being able to contact the SILKNOW project	Via email, or a contact form	All	All	Yes

Table 7. List of the functionalities that ADASilk should fulfil according to SSH partners, and their relation to personas and sectors that were identified in D2.4. The last column, in grey, summarizes the outcomes of the functionality evaluation.

3.2.1. Functionality 01

Description of the functionality: Having a personal account.

Fulfilled: Yes.

Implementation/use: Users can sign into ADASilk via different options, namely, Google, Facebook and Twitter. An example is shown in Figure 20, where a user has access to the logging options after clicking the button "PROFILE".

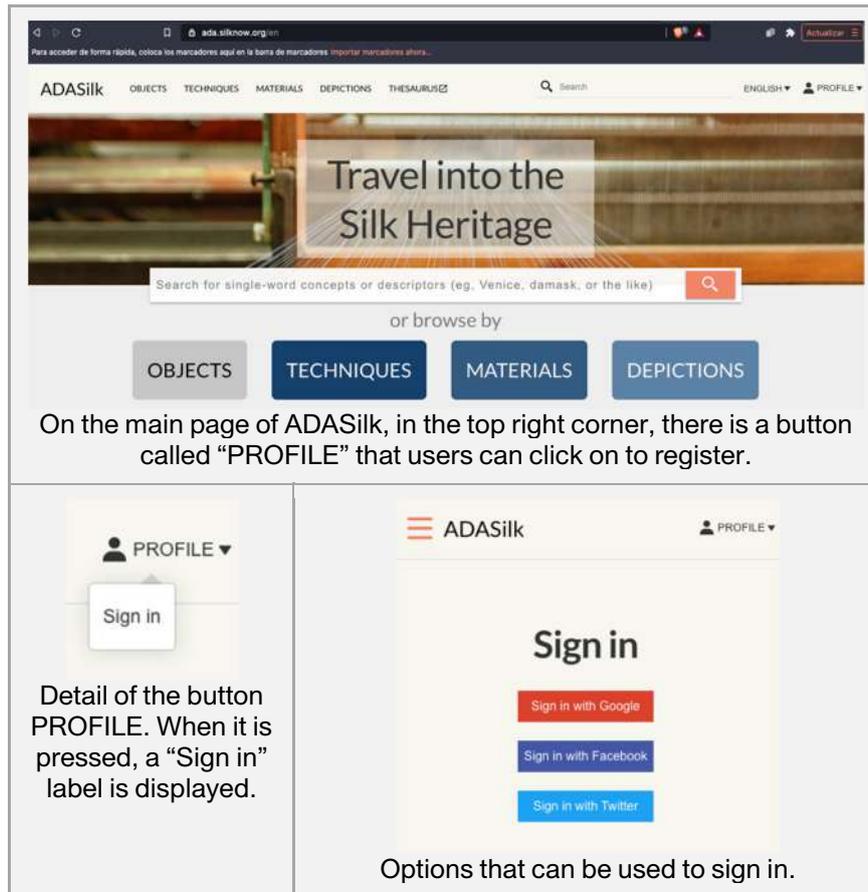


Figure 20. Screenshots of ADASilk to test functionality 01.

3.2.2. Functionality 02

Description of the functionality: Being able to save favorite images.

Fulfilled: Yes.

Implementation/use: When displaying individual objects in ADASilk users can select to download any images they wish.

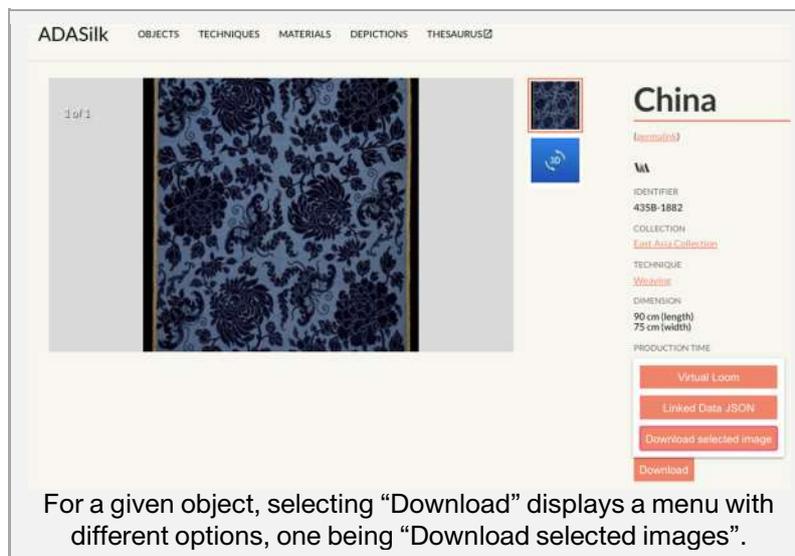


Figure 21. Screenshots of ADASilk to test functionality 02.

3.2.3. Functionality 03

Description of the functionality: Being able to create and share a personal selection of items, a kind of curated exhibition.

Fulfilled: Yes.

Implementation/use: When users are registered in ADASilk, they can create their own list of objects and then share them through different media such as Facebook, Twitter, Whatsapp, LinkedIn, email, etc.

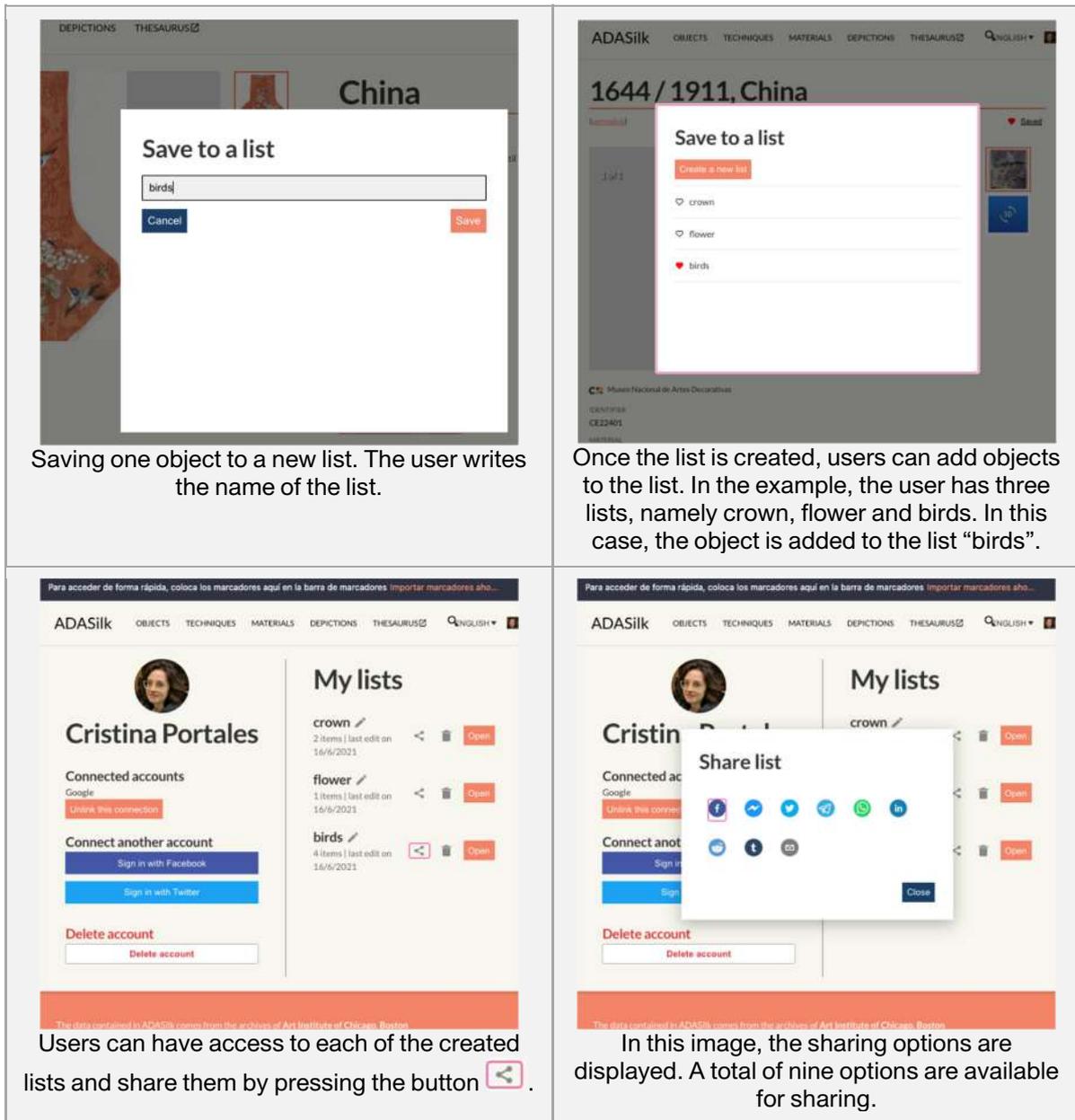


Figure 22. Screenshots of ADASilk to test functionality 03.

3.2.4. Functionality 04

Description of the functionality: Having access to information about the latest updates and the new materials.

Fulfilled: Not yet implemented.

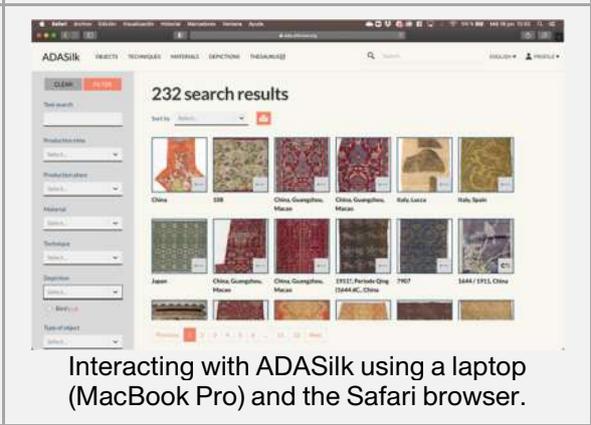
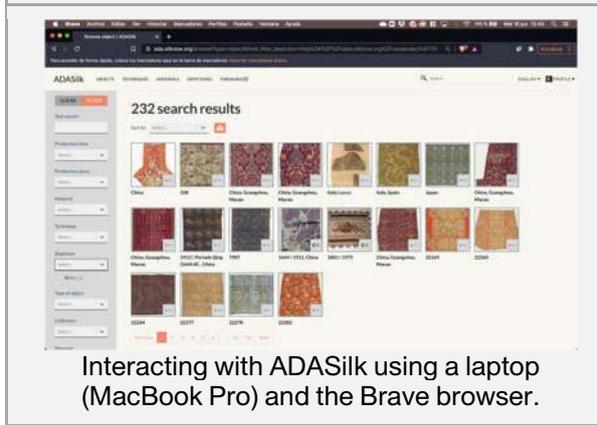
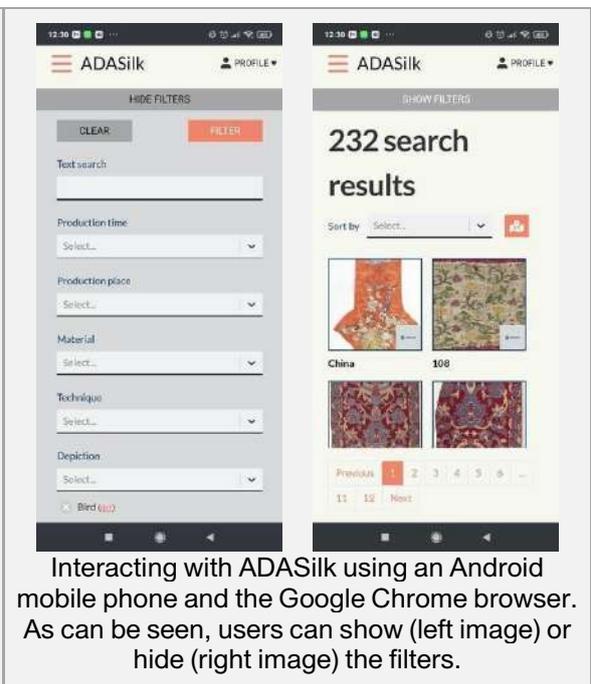
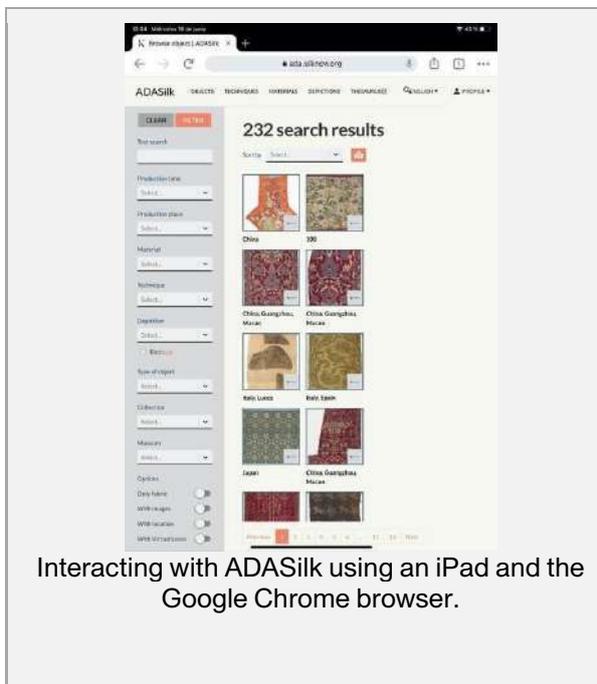
Implementation/use: Although this functionality has not yet been implemented, it is feasible to do so in the scope of the project. To this end, we will integrate a “History” menu into ADASilk showing when the various museums have been gradually added to the KG.

3.2.5. Functionality 05

Description of the functionality: Making all our web resources responsive to all user devices (desktop/laptop, tablet, smartphone).

Fulfilled: Yes.

Implementation/use: ADASilk is prepared to be used on standard devices (desktop/laptop, tablet, smartphone) and making use of most browsers (Chrome, Safari, Edge, etc.). A few examples are shown in Figure 13, where the same action, filtering by depiction/birds, is applied on different devices and browsers, also involving different operating systems.



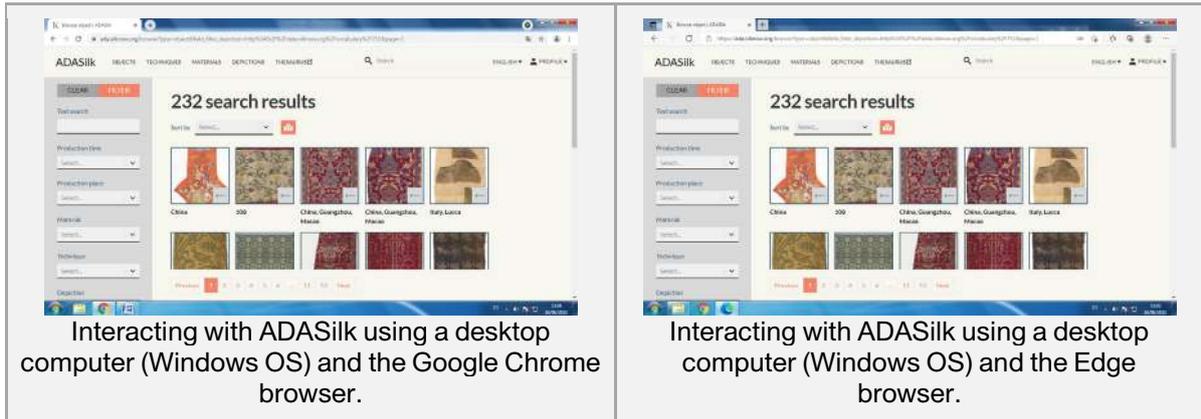


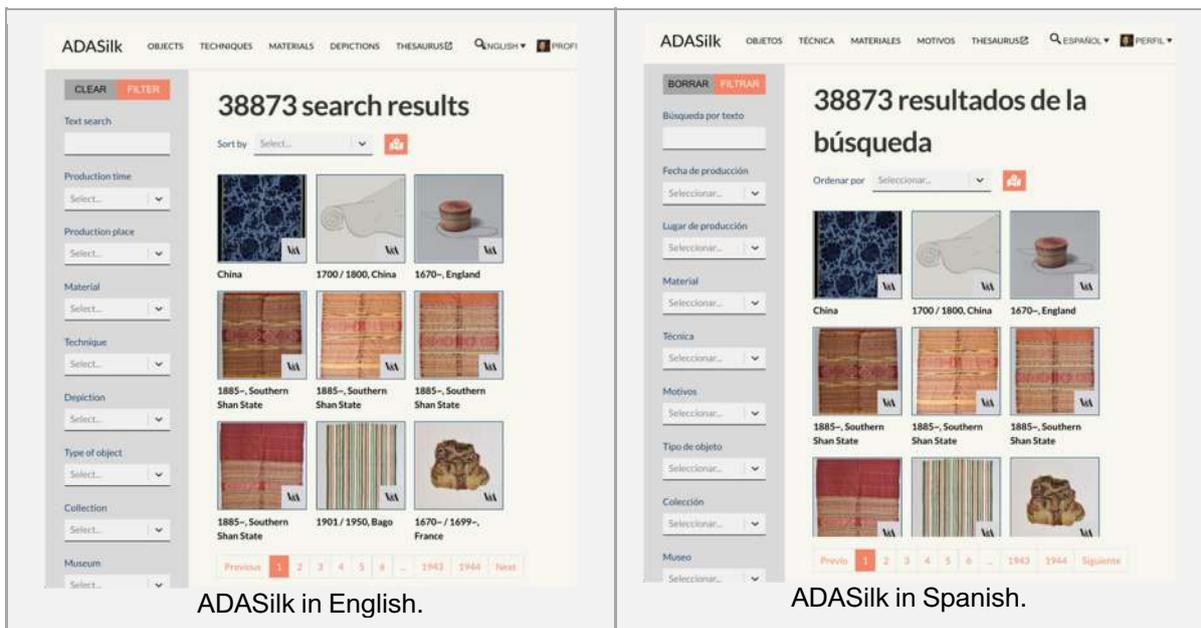
Figure 23. Screenshots of ADASilk to test functionality 05.

3.2.6. Functionality 06

Description of the functionality: Choosing the language.

Fulfilled: Yes.

Implementation/use: By default, ADASilk's user interface is shown in English; however, Spanish, French and Italian are also integrated so users can choose how to see the interface menus. An example is shown in Figure 24.



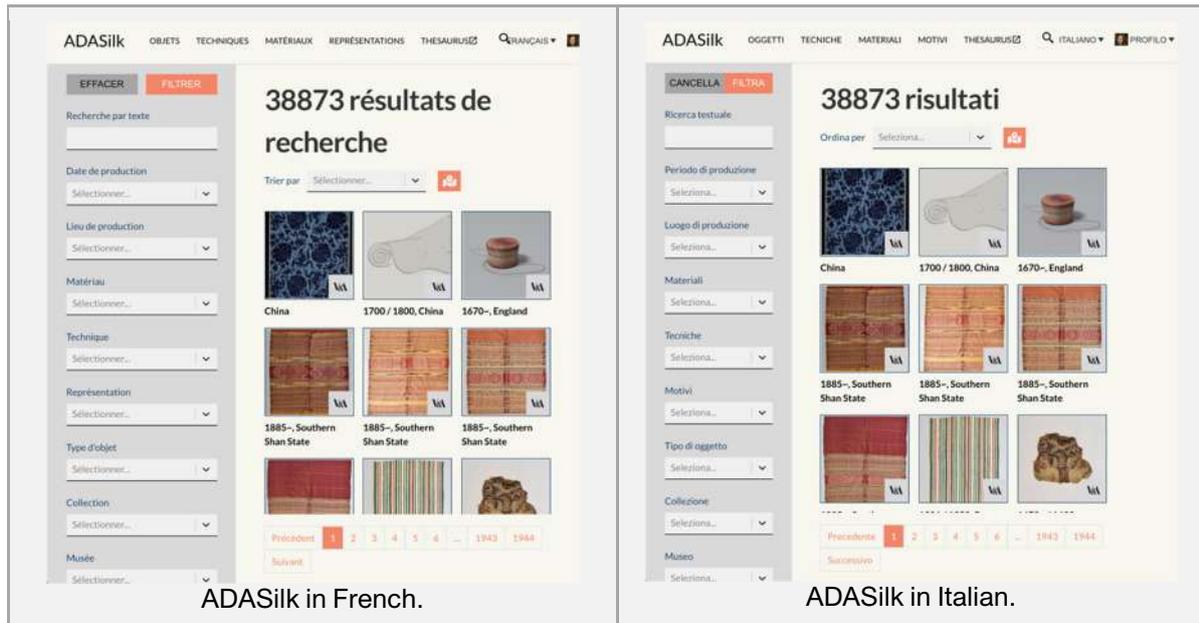


Figure 24. Screenshots of ADASilk to test functionality 06.

3.2.7. Functionality 07

Description of the functionality: Using a simple search interface.

Fulfilled: Yes.

Implementation/use: Text-based filtering is integrated into ADASilk, so users can type a free text. While the user is typing, the system remembers previous searches and also delivers results in real time. This is depicted in Figure 25.

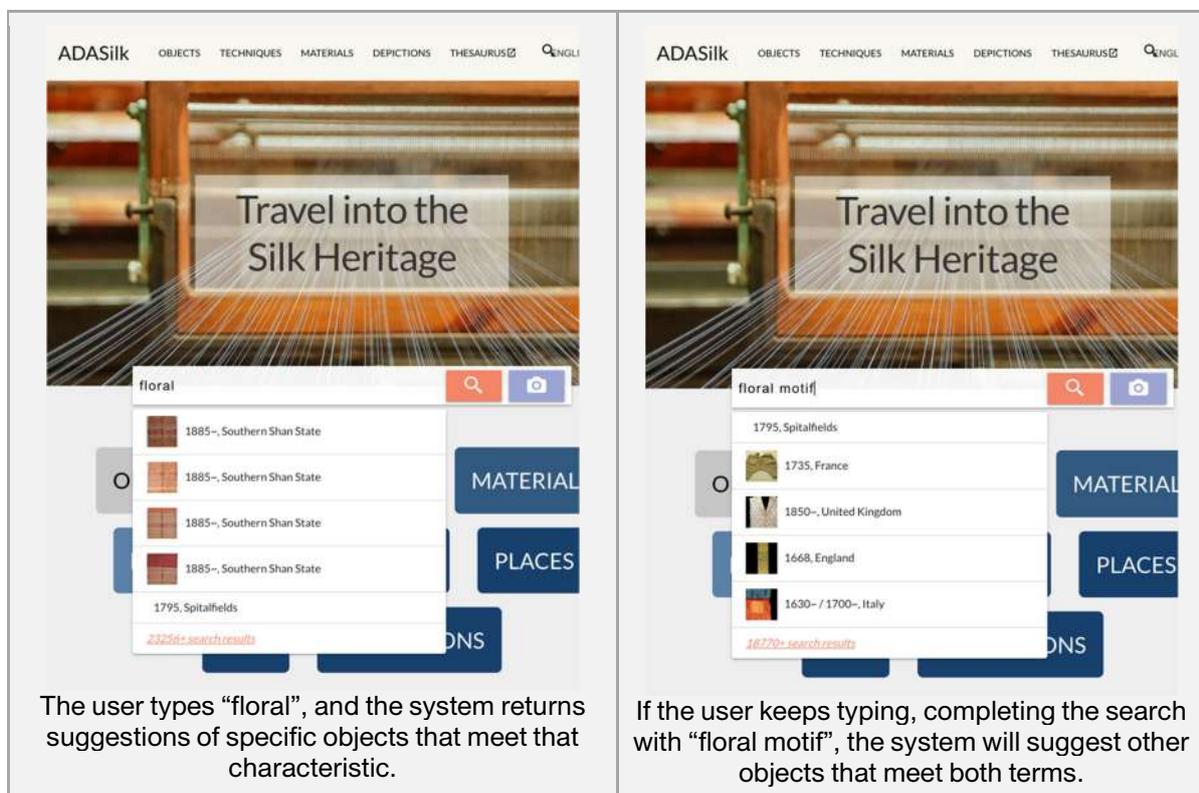


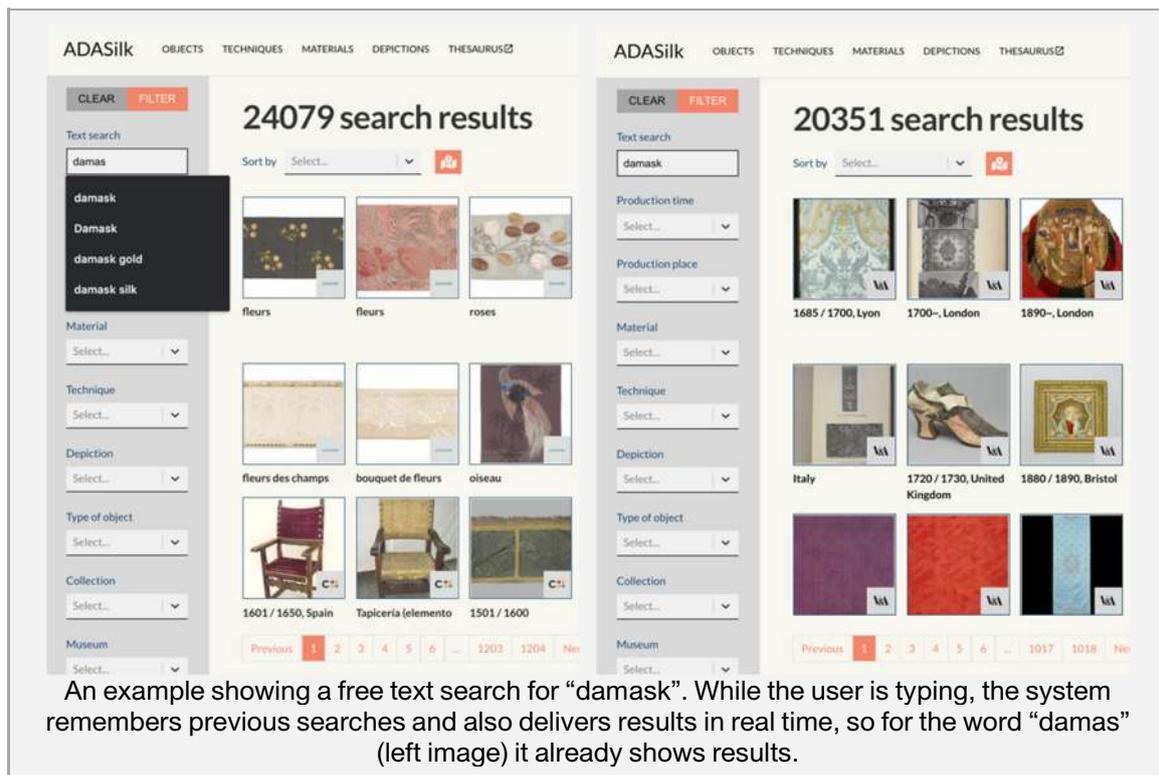
Figure 25. Screenshots of ADASilk to test functionality 07.

3.2.8. Functionality 08

Description of the functionality: Using an advanced search interface and filtering results with facets.

Fulfilled: Yes.

Implementation/use: On the OBJECTS page of ADASilk, text-based filtering is integrated so users can type a free text. While the user is typing, the system remembers previous searches and also delivers results in real time.



An example showing a free text search for “damask”. While the user is typing, the system remembers previous searches and also delivers results in real time, so for the word “damas” (left image) it already shows results.

Figure 26. Screenshots of ADASilk to test functionality 08.

3.2.9. Functionality 09

Description of the functionality: Using an advanced search interface and filtering results with facets: historical period and/or geographical origin.

Fulfilled: Yes.

Implementation/use: In ADASilk’s filtering options, the fields “Production time” and “Production place” are considered. For the former, the time is divided into centuries. While for the latter, the place has different granularities, from continents to countries or regions. For each field, one or different options can be selected. Selecting different options for a single field results in an OR operation, whilst selecting different filters results in an AND operation. Examples are shown in Figure 27.

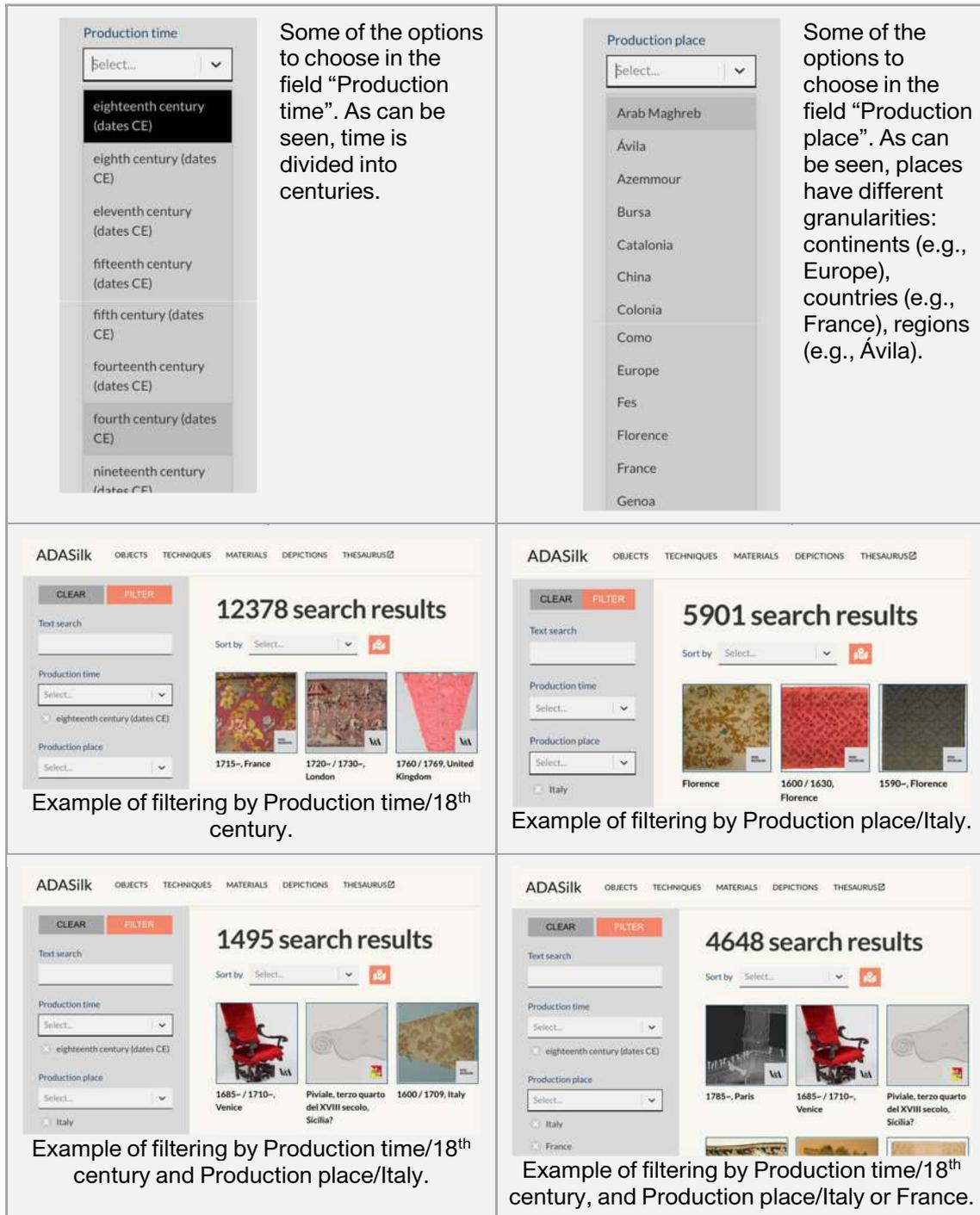


Figure 27. Screenshots of ADASilk to test functionality 09.

3.2.10. Functionality 10

Description of the functionality: Using an advanced search interface and filtering results with facets: weaving techniques and fabrics.

Fulfilled: Yes.

Implementation/use: In ADASilk’s filtering options, there is the field “Technique” which allows users to filter by a selected technique. On the one hand, to filter objects by fabric, two options are implemented. On the other hand, users can filter by “Type of objects/fabrics”.

SILKNOW

There is also a specific tab option that indicates “only fabric” to be considered in the graphical interface. Examples are shown in Figure 18. Like in other filters, selecting different options for a single field results in an OR operation, while selecting different filters results in an AND operation.

	<p>Some of the techniques that users can select in the field “Technique”.</p>		<p>Selecting “fabrics” in the field “Type of object”.</p>	<p>Interface for the option to filter by “only fabric”.</p>
<p>Example of filtering by type of object/fabrics.</p>		<p>Example of filtering by “only fabric”.</p>		

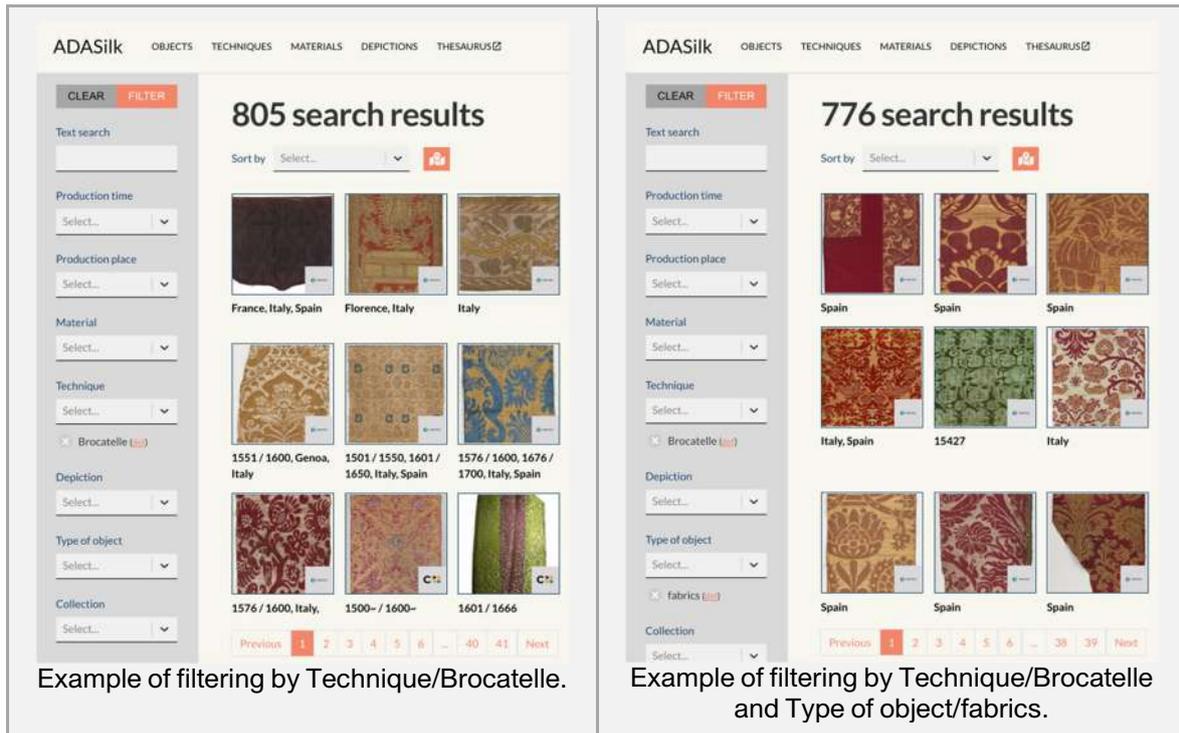


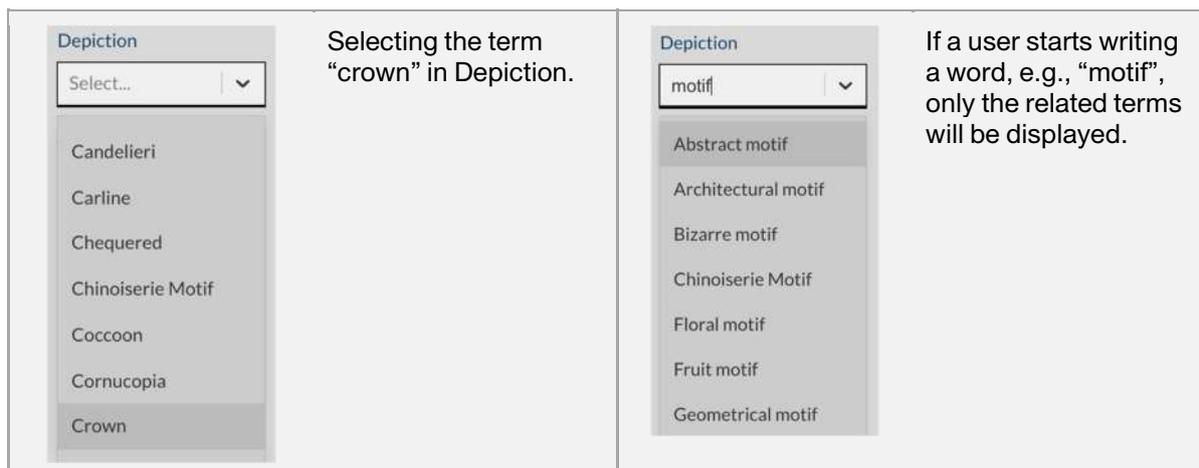
Figure 28. Screenshots of ADASilk to test functionality 10.

3.2.11. Functionality 11

Description of the functionality: Using an advanced search interface and filtering results with facets: motifs, decorative patterns.

Fulfilled: Yes.

Implementation/use: Users can filter by motives and/or decorative patterns with “Depiction”. Examples are shown in Figure 29.



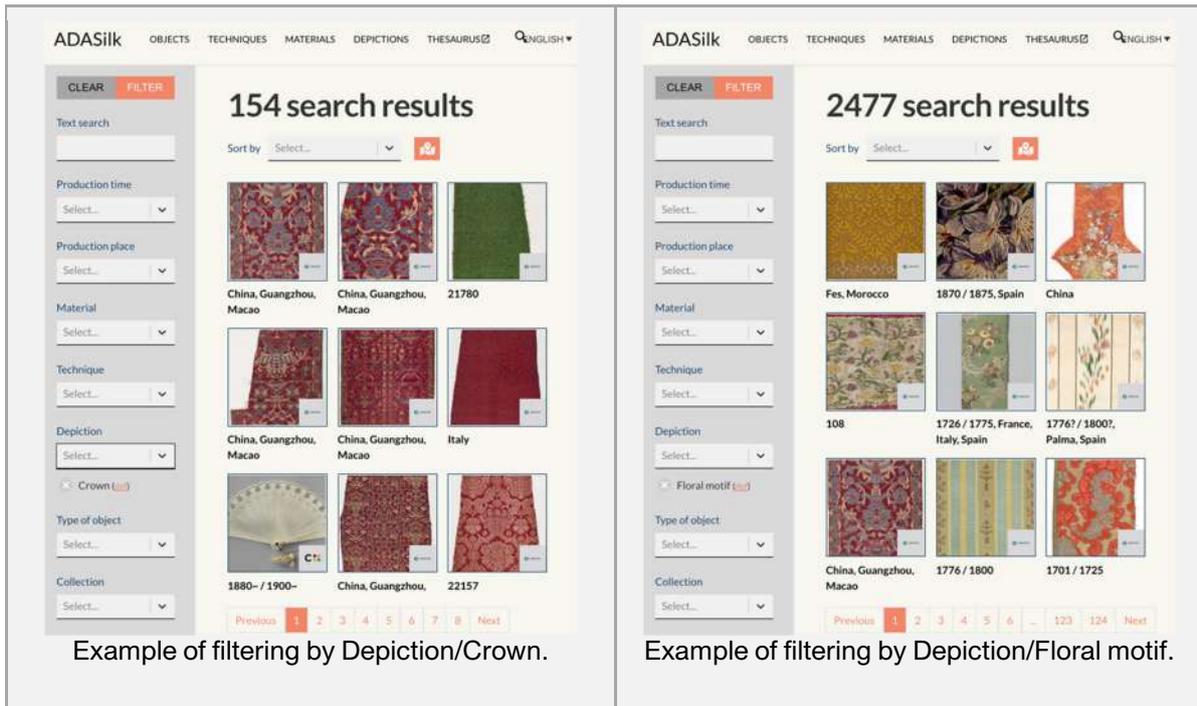


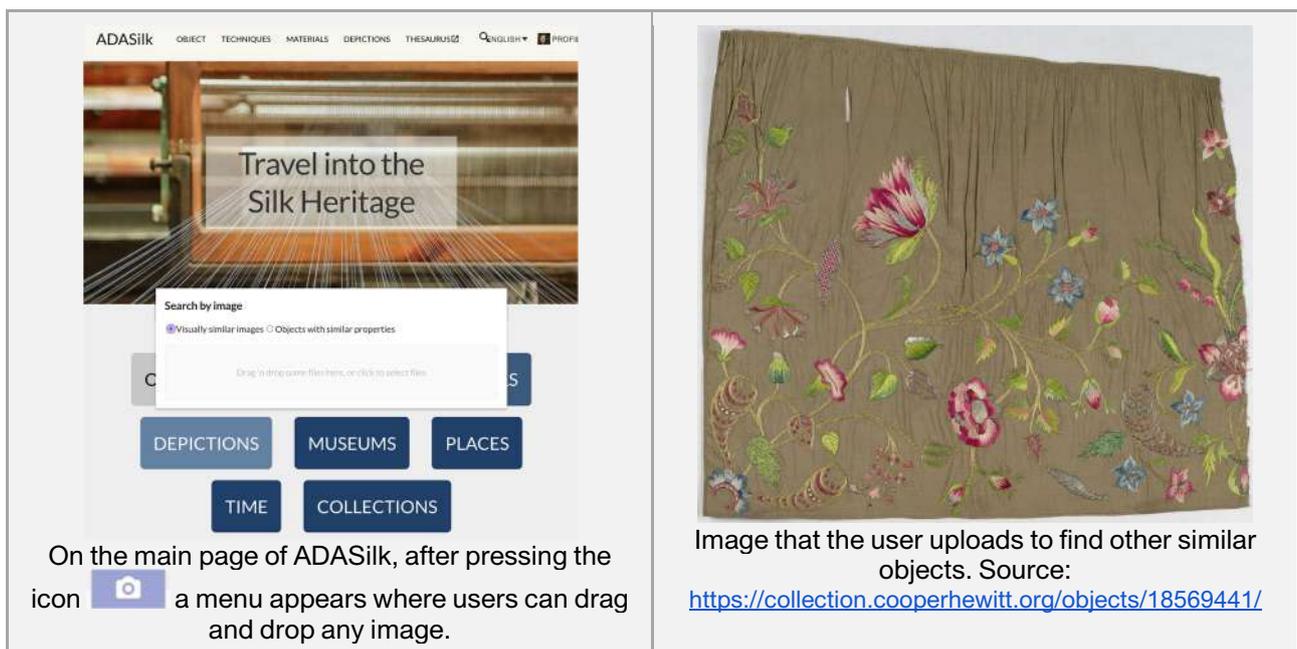
Figure 29. Screenshots of ADASilk to test functionality 11.

3.2.12. Functionality 12

Description of the functionality: Make a query based on a single image uploaded by the user, or one selected in the repository. Give the option to include metadata analysis (as all of the above) or to search only by visual similarity.

Fulfilled: Yes.

Implementation/use: From the main page of ADASilk, the user can upload an image of choice and carry out a search by image. She/he can ask for either visually similar images or objects with similar properties. This is shown in Figure 30.



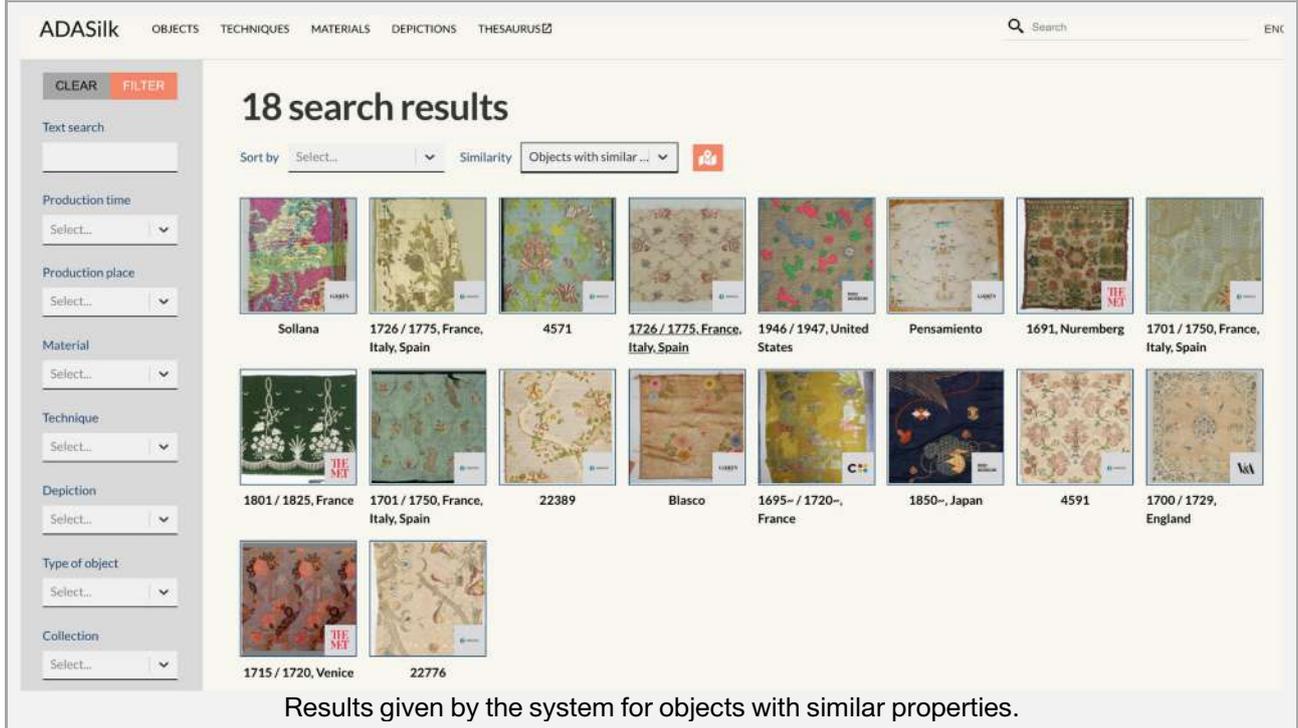
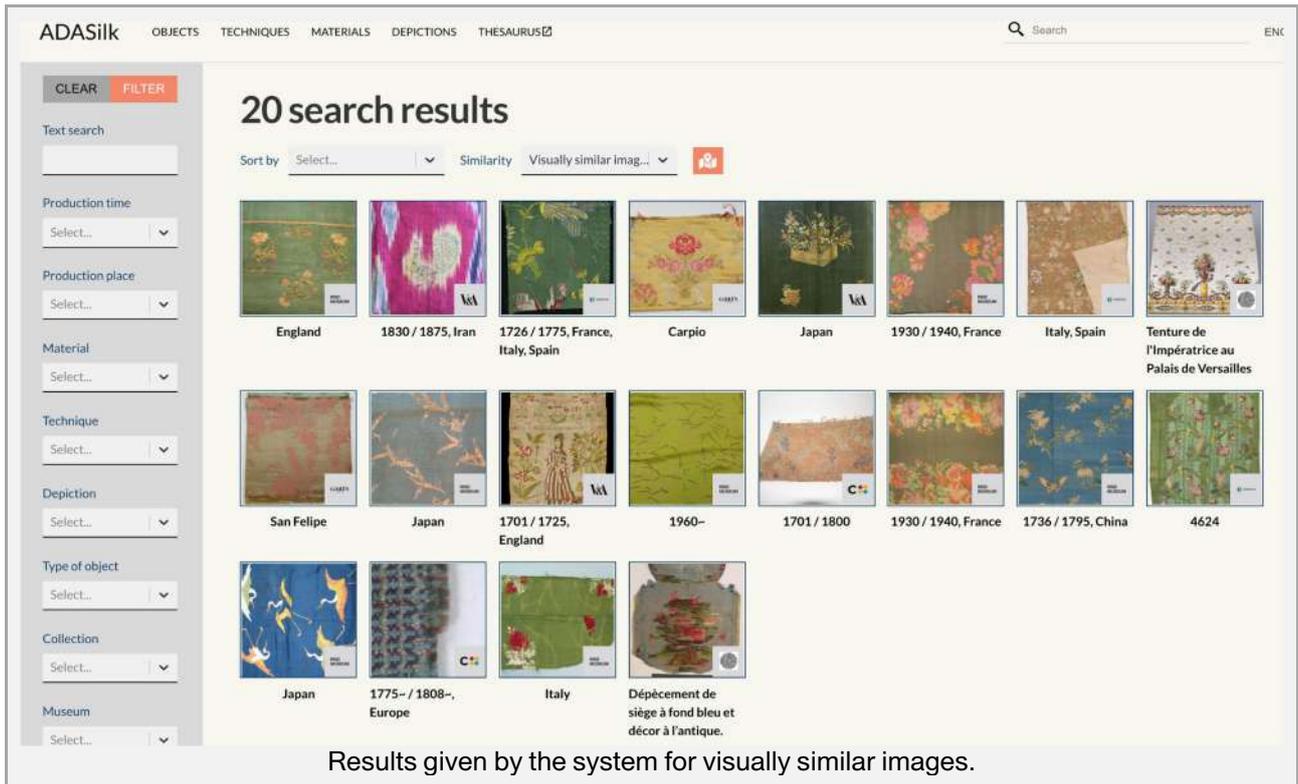


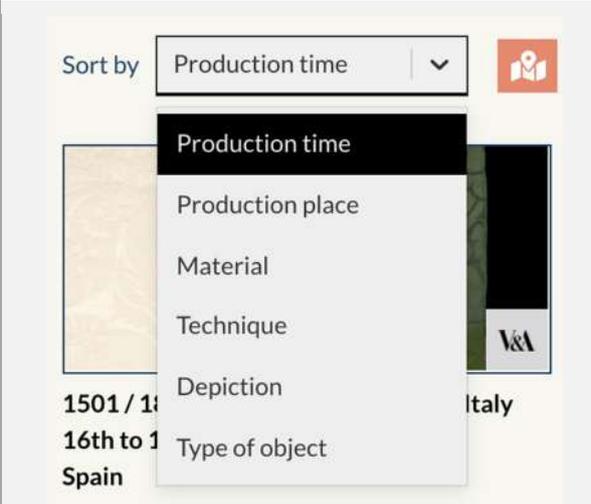
Figure 30. Screenshots of ADASilk to test functionality 12.

3.2.13. Functionality 13

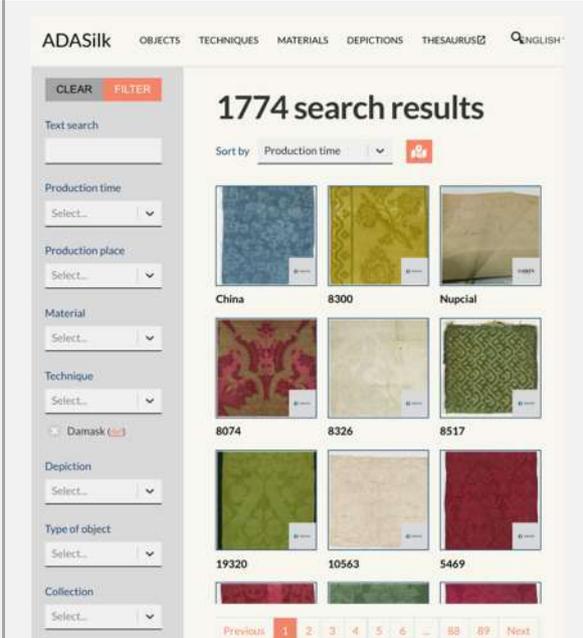
Description of the functionality: Sorting the results by field (alphabetical order for text fields, ascending or descending order for numerical or date fields) and also with facets (such as “by country”).

Fulfilled: Yes.

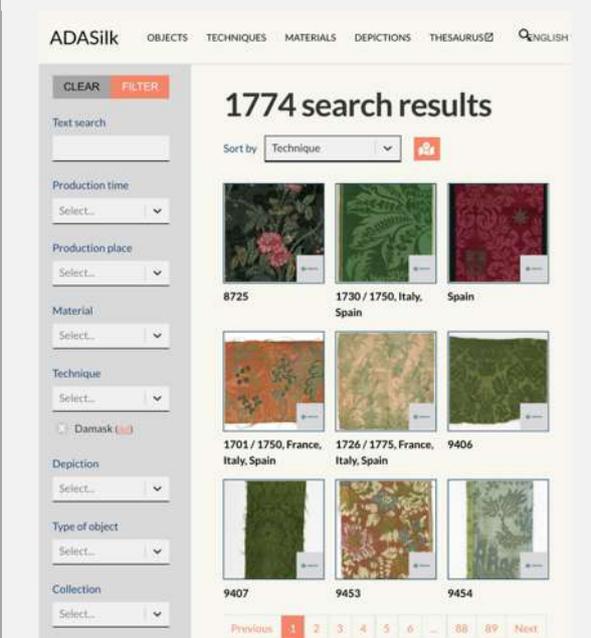
Implementation/use: The terms included in each facet are sorted by order. Examples of this have been shown in e.g., Figure 19. After filters are applied, the results can be sorted by production time, production place, material, technique, depiction or type of object. Examples can be seen in Figure 31.



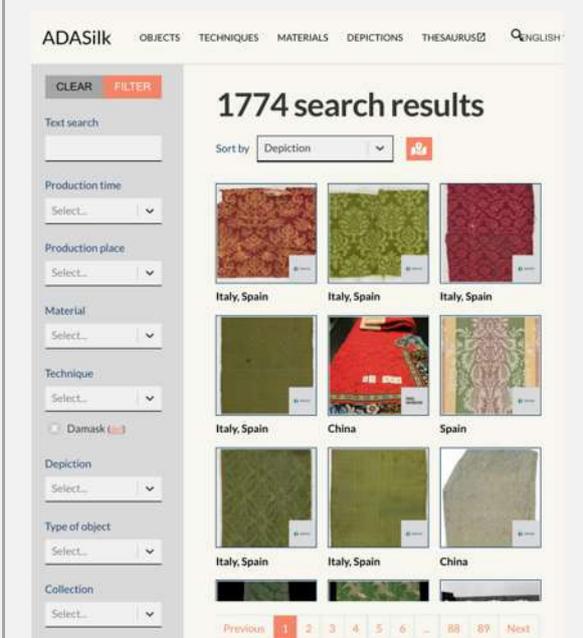
Options implemented in ADASilk in order to sort the results.



Example of sorting by Production time for the search Technique/Damask.



Example of sorting by Technique for the search Technique/Damask.



Example of sorting by Depiction for the search Technique/Damask.

Figure 31. Screenshots of ADASilk to test functionality 13.

3.2.14. Functionality 14

Description of the functionality: Using a zoom tool for visualization.

Fulfilled: Yes.

Implementation/use: When accessing a single object that has at least one image, users can open the image in a new window and apply zoom “+” and “-”. This is exemplified in Figure 32.

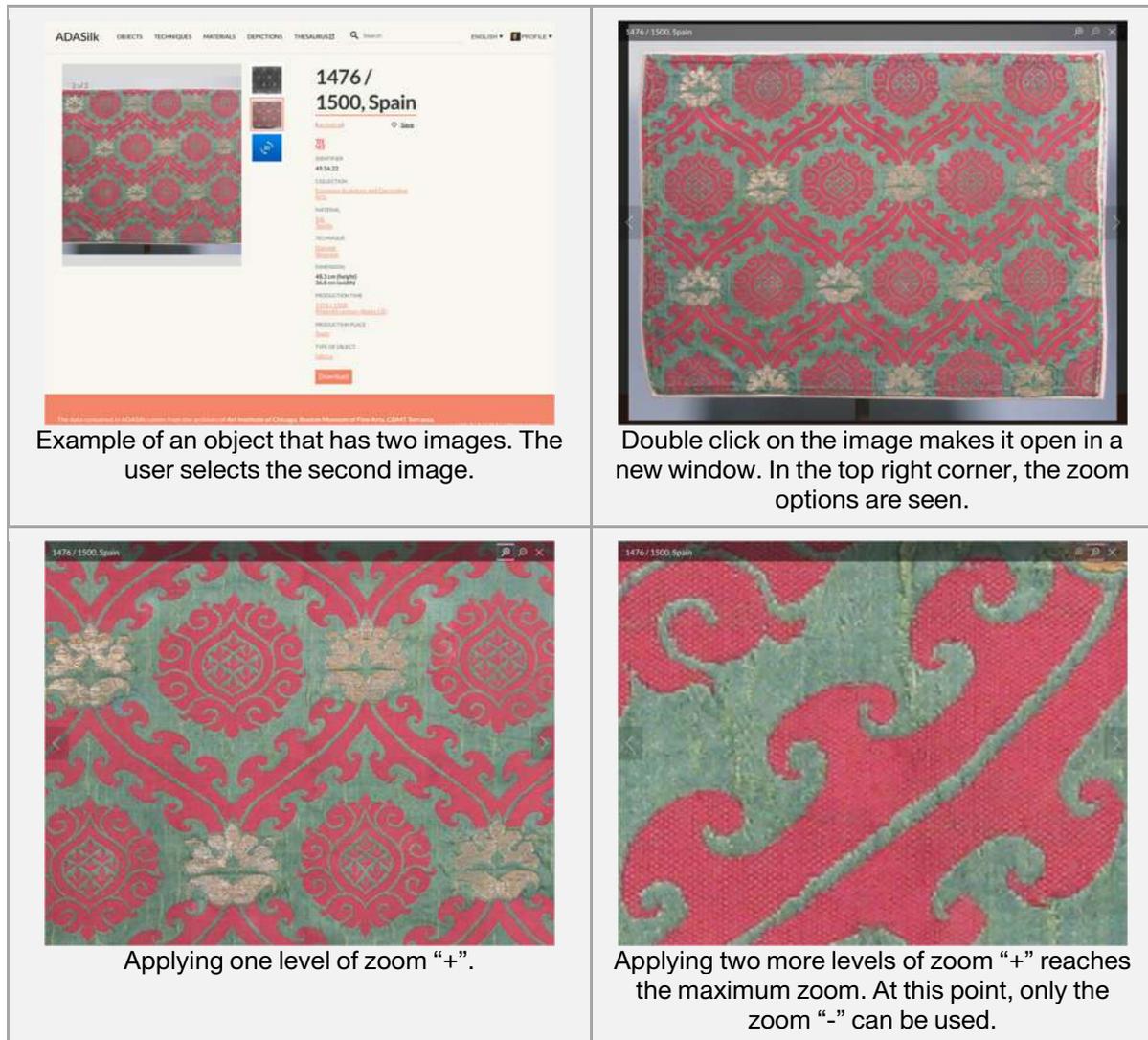


Figure 32. Screenshots of ADASilk to test functionality 14.

3.2.15. Functionality 15

Description of the functionality: Provide the user with a link (when available) back to the online catalogue of the owning institution to get more information, etc.

Fulfilled: Yes.

Implementation/use: When entering a single object in the description, ADASilk provides the link to the website of the owning institution. A pair of examples are given in Figure 33.

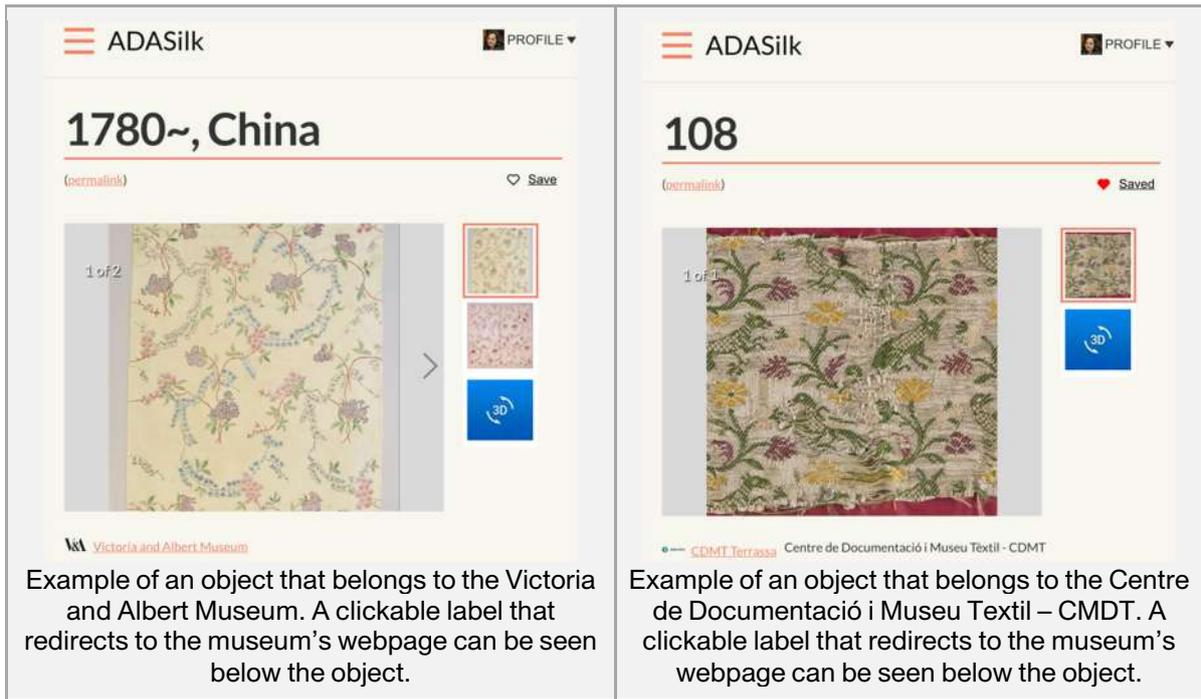


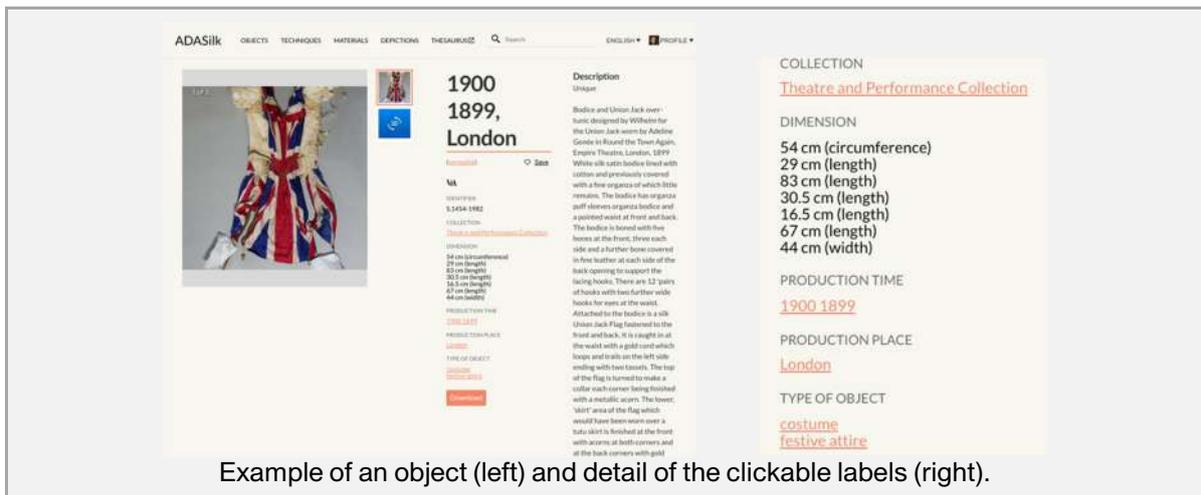
Figure 33. Screenshots of ADASilk to test functionality 15.

3.2.16. Functionality 16

Description of the functionality: Being able to find other SILKNOW related records by clicking on keywords or CHIs or authors, etc.

Fulfilled: Yes.

Implementation/use: For a given object, a set of clickable labels are listed that lead users to related objects. Examples are shown in Figure 34.



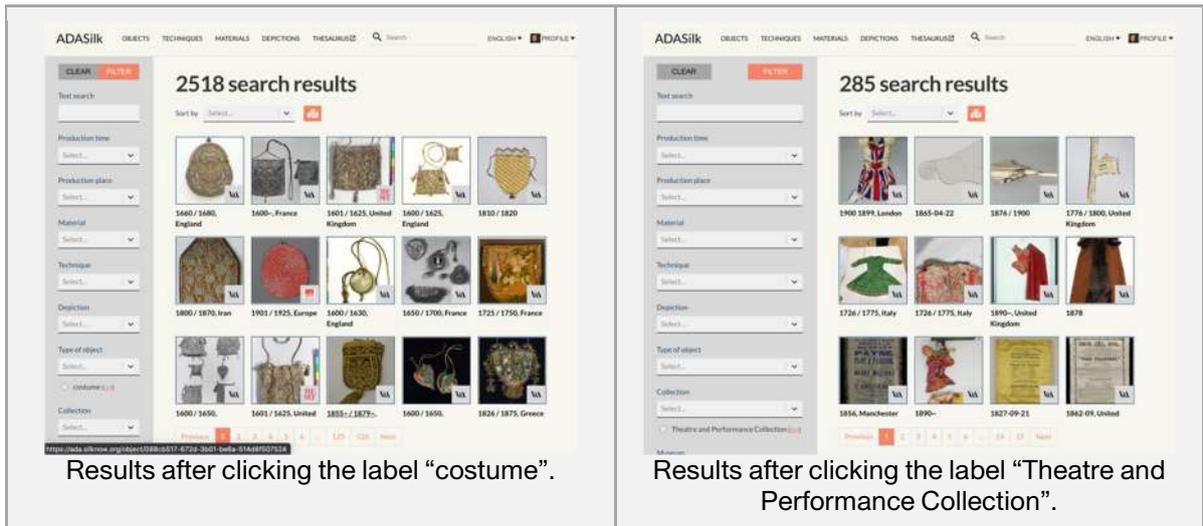


Figure 34. Screenshots of ADASilk to test functionality 16.

3.2.17. Functionality 17

Description of the functionality: Being able to find other SILKNOW records of related objects (clickable list of related records if this information was included in the original records).

Fulfilled: Yes.

Implementation/use: ADASilk implements the possibility of looking for related objects which have visually similar images, or objects with similar properties. An example of how to do this is given in Figure 35.

Visually similar images

Objects with similar properties

View similar

For a given object, the user can press the button "View similar", and two options appear; so he/she can choose between visually similar images or objects with similar properties.

1725~, 1735

Art Institute of Chicago Art Institute of Chicago (ARTIC)

IDENTIFIER
1961.553

MATERIAL
[Animal fibre](#) 98%

[Metal thread](#) 4%

TECHNIQUE
[Embroidery](#) 55%

DEPICTION

An example of an object for which the user wants to see visually similar images and/or objects with similar properties.

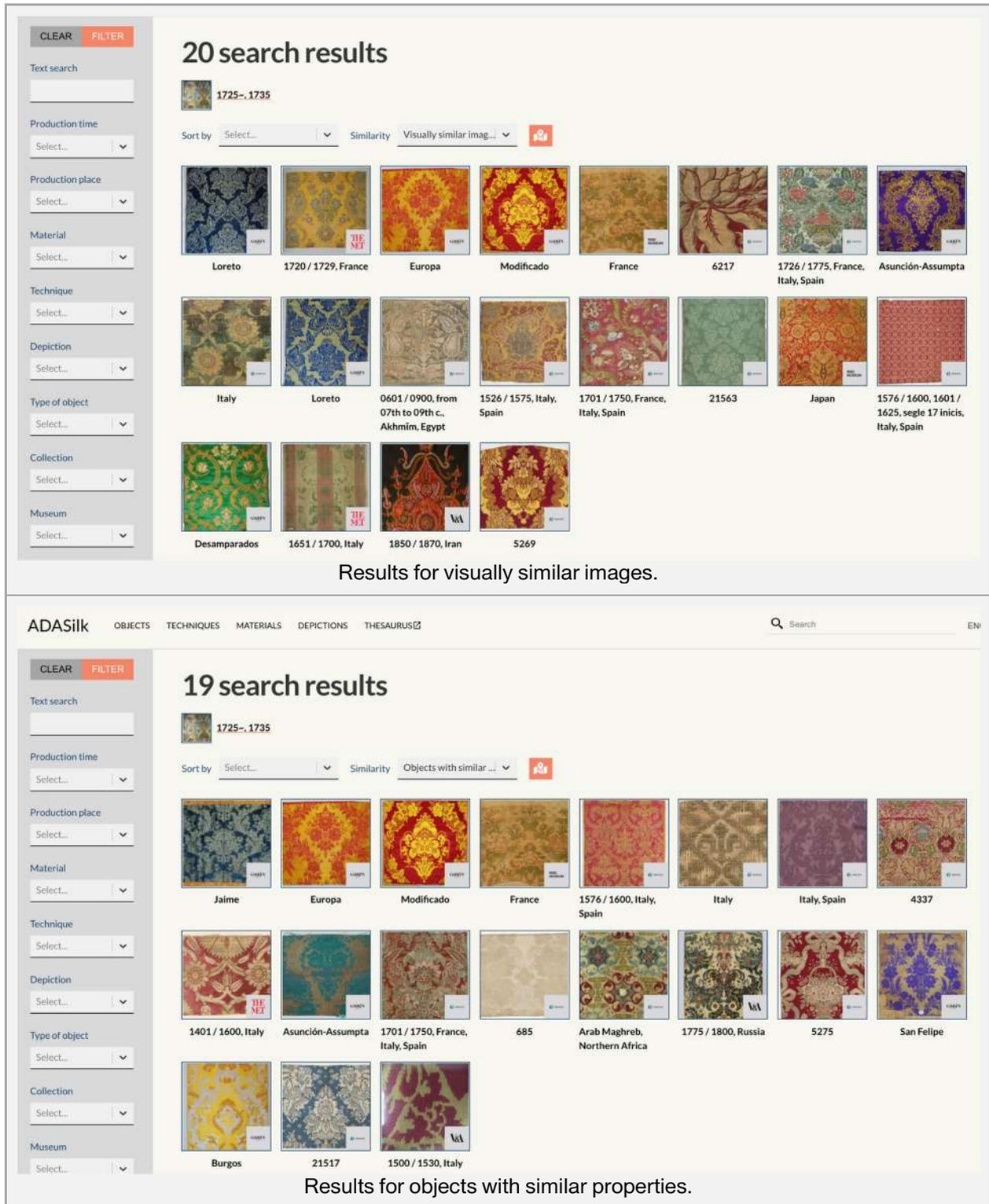


Figure 35. Screenshots of ADASilk to test functionality 17.

3.2.18. Functionality 18

Description of the functionality: Having access to the name of the owning institution in the record and being able to click on the names for more information (location, contacts, etc.).

Fulfilled: Not yet implemented.

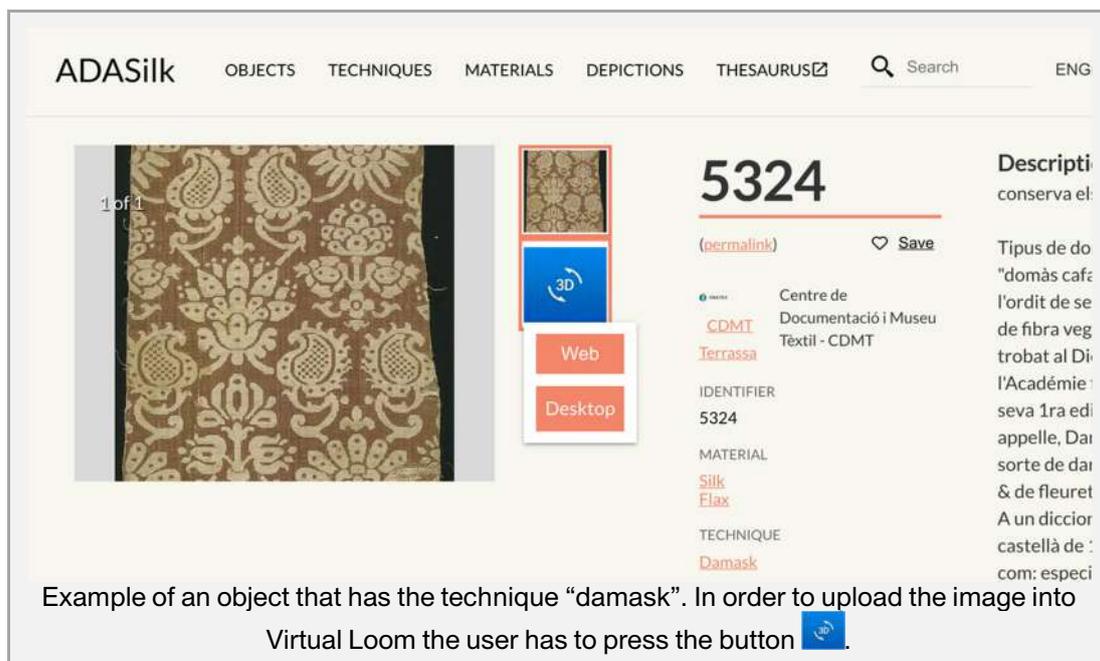
Implementation/use: This functionality will be implemented in the scope of the project. To this end, we will include the contact details of the owning institution, apart from the link to their online catalogue (refer to functionality 15).

3.2.19. Functionality 19

Description of the functionality: Clicking on a weaving technique in order to access a detailed analysis of it in the Virtual Loom, when available.

Fulfilled: Yes.

Implementation/use: Any ADASilk object that has an image can be automatically uploaded to Virtual Loom. Once in Virtual Loom, the tool suggests a technique (the actual technique of the object, if known), so the user can select it if it is integrated into Virtual Loom. This is done in this way because, on the one hand, Virtual Loom has a limited number of weaving techniques compared to all the possible techniques and variations and, on the other hand, any image can be woven using any of the implemented techniques. Therefore, we do not limit the use of Virtual Loom to only those fabrics that have a technique integrated into Virtual Loom. An example is given in Figure 36.



Example of an object that has the technique “damask”. In order to upload the image into Virtual Loom the user has to press the button .

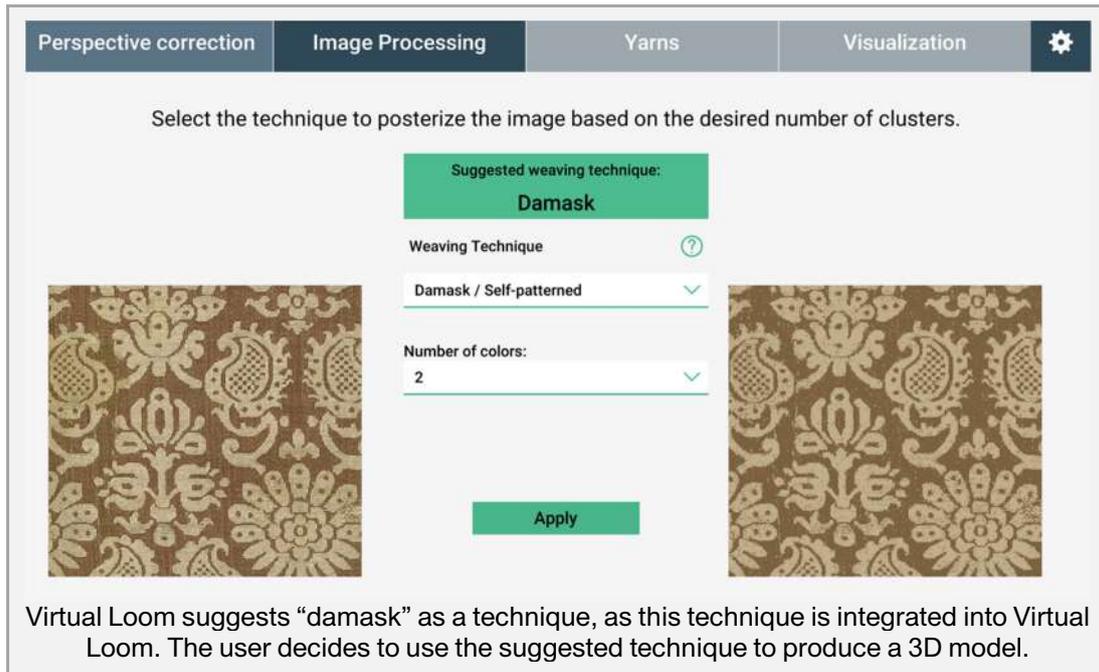


Figure 36. Screenshots of ADASilk to test functionality 19.

3.2.20. Functionality 20

Description of the functionality: Querying our database through an API, not just through a web interface.

Fulfilled: Yes.

Implementation/use: ADASilk deploys a public RESTful API. An external application can query the Knowledge Graph through this API, without authentication. In order to check this functionality, it is only necessary to invoke the API through a REST client and process the JSON file returned. As an example, we use a query to get the results of all the pieces created in Italy. This query is:

https://grlc.eurecom.fr/api-git/silknow/api/obj_list?location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql

This query is typed as a GET command in the Mozilla extension RESTClient (Figure 37), and the response contains a JSON file with the same results as the web user interface.

The command generated by RESTClient is the following:

```
curl -X GET -k -i 'https://grlc.eurecom.fr/api-git/silknow/api/obj_list?location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql'
```

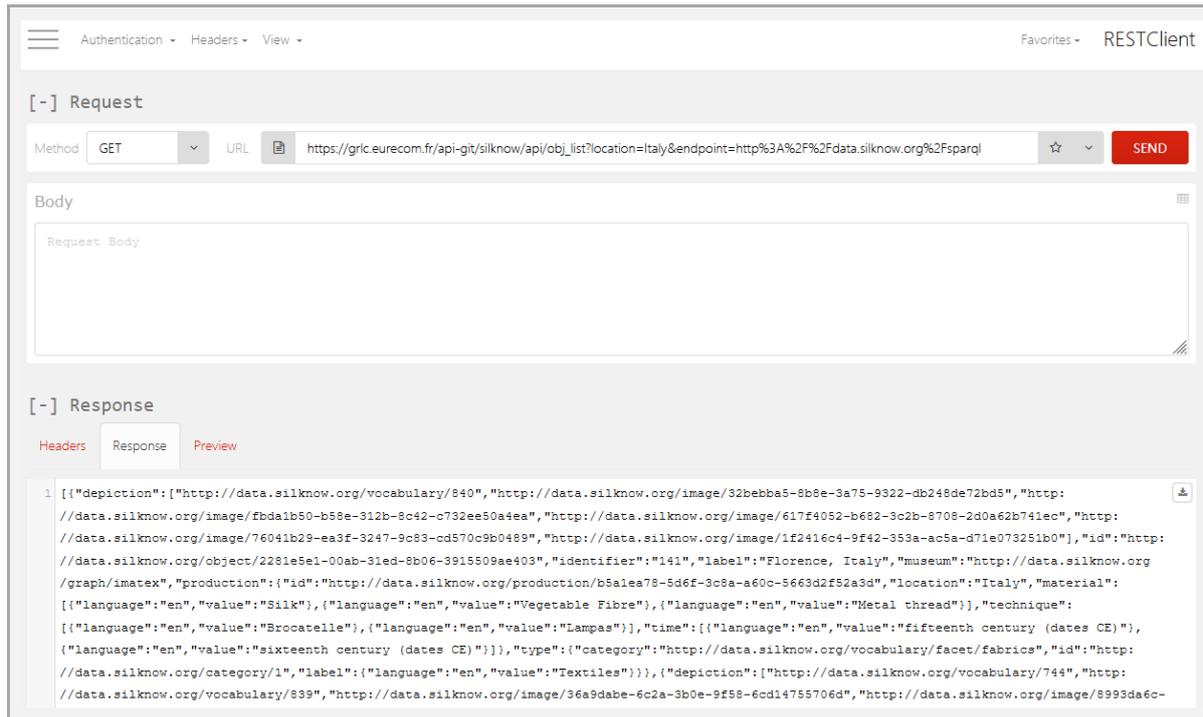


Figure 37. Screenshots of ADASilk to test functionality 20.

3.2.21. Functionality 21

Description of the functionality: Downloading a list of selected results with a standard, basic set of metadata.

Fulfilled: Yes.

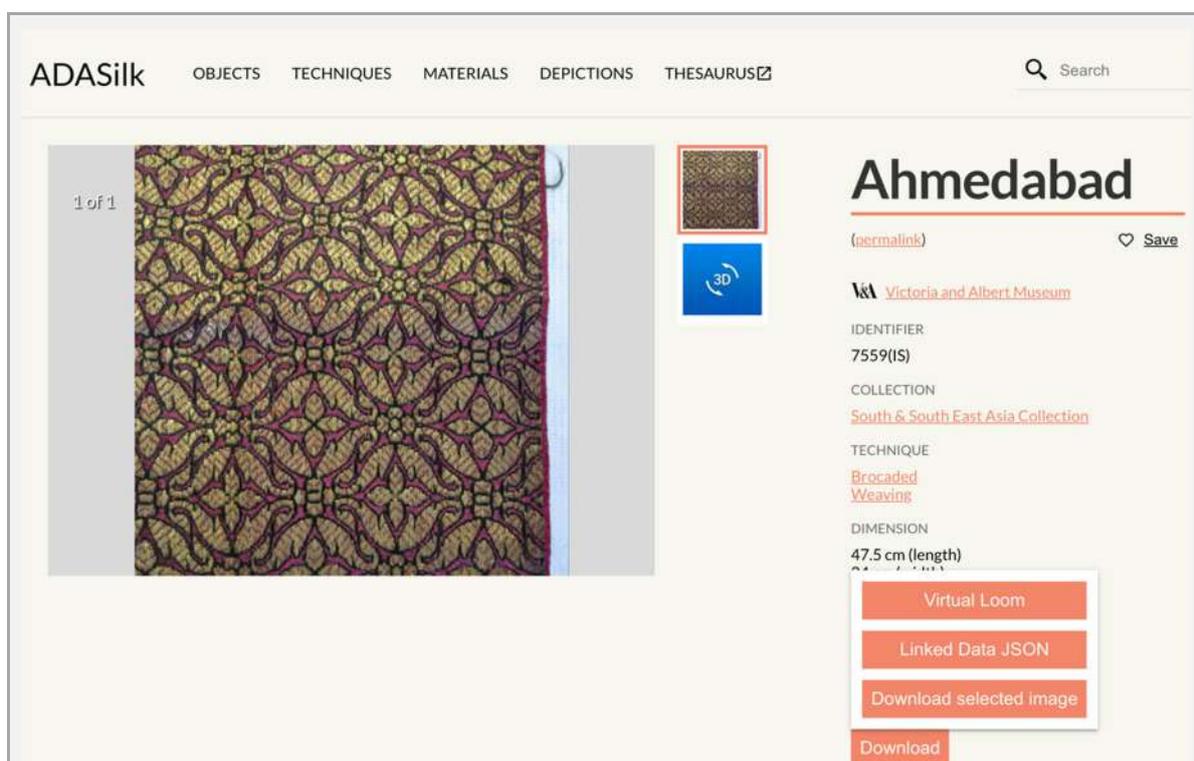
Implementation/use: Users who are registered in ADASilk can create lists of objects (recall functionality 03), which can be downloaded. This is exemplified in Figure 38, where a user has a list called “birds” which contains four objects and downloads the list.

3.2.22. Functionality 22

Description of the functionality: Downloading individual search results (one record) in the format chosen by the user: image only in bitmap format (jpg, png, etc.), metadata only, entire record (pdf, csv, rtf), etc.

Fulfilled: Yes.

Implementation/use: For each individual object integrated into ADASilk, users can choose between different options for downloading the related information, namely: “Virtual Loom”, which downloads a JSON file that contains the information relevant for Virtual Loom; “Linked Data JSON”, which downloads a JSON file with all the information related to said object; and “Download selected image”, which downloads the selected image in JPEG format. An example is given in Figure 39.



Example of an object in ADASilk and the downloading options that are available for each individual object integrated into ADASilk.

```
{
  "language": "EN",
  "imgUri": "https://silknow.org/silknow/media/vam/O478172_0.jpg",
  "dimension": {},
  "technique": [
    "Brocaded",
    "Weaving"
  ],
  "weaving": "Plain",
  "background": {
    "color": {
      "r": 0.7075471878051758,
      "g": 0.2302865833044052,
      "b": 0.2302865833044052,
      "a": 0
    },
    "materials": [
      null
    ],
    "endpoint": "http://grc.eurecom.fr/api-git/silknow/api/",
    "analytics": false
  }
}
```

Downloaded JSON file that contains the information that is relevant for Virtual Loom.

```
{
  "@type": "http://erlangen-crm.org/current/E22_Made-Object",
  "@id": "http://data.silknow.org/object/a31f4190-e859-3a43-af3e-0b8c183e350e",
  "@graph": "http://data.silknow.org/graph/vam",
  "identifier": "7559(IS)",
  "representation": {
    "@id": "http://data.silknow.org/image/4bbae8ac-98a0-3da6-b986-29772daebb6b",
    "image": "https://silknow.org/silknow/media/vam/O478172_0.jpg",
    "legalBody": {},
    "composed": {
      "@type": "http://erlangen-crm.org/current/E78_Collection",
      "@id": "http://data.silknow.org/collection/ed3dbc7c-4540-320d-a055-6841c02cc217",
      "@type": "http://erlangen-crm.org/current/E78_Collection",
      "label": "South & South East Asia Collection",
      "material": {},
      "__material": {},
      "technique": [
        {
          "@id": "http://data.silknow.org/vocabulary/192",
          "label": "Brocaded"
        },
        {
          "@id": "http://data.silknow.org/vocabulary/526",
          "label": "Weaving"
        }
      ],
      "usedType": {},
      "depiction": {},
      "dimension": {
        "@id": "http://data.silknow.org/object/a31f4190-e859-3a43-af3e-0b8c183e350e/dimension/1",
        "type": "length",
        "value": 47.5,
        "unit": "cm"
      }
    }
  }
}
```



Figure 39. Screenshots of ADASilk to test functionality 22.

3.2.23. Functionality 23

Description of the functionality: Being able to share content (one single record) on social networks (e.g., Facebook, Instagram, Pinterest), by email, with a URL.

Fulfilled: Yes.

Implementation/use: As explained in functionality 03, when users are registered in ADASilk they can share content through different media such as Facebook, Twitter, Whatsapp, LinkedIn, email, etc. This was shown in Figure 12.

3.2.24. Functionality 24

Description of the functionality: Being able to share multiple content: (a query, a user selection of records, etc) using the same procedures as above (social media, email, URL, etc.).

Fulfilled: Yes.

Implementation/use: The search query in ADASilk generates a permalink, so anyone can share the URL to anyone else and they will get the same results page.

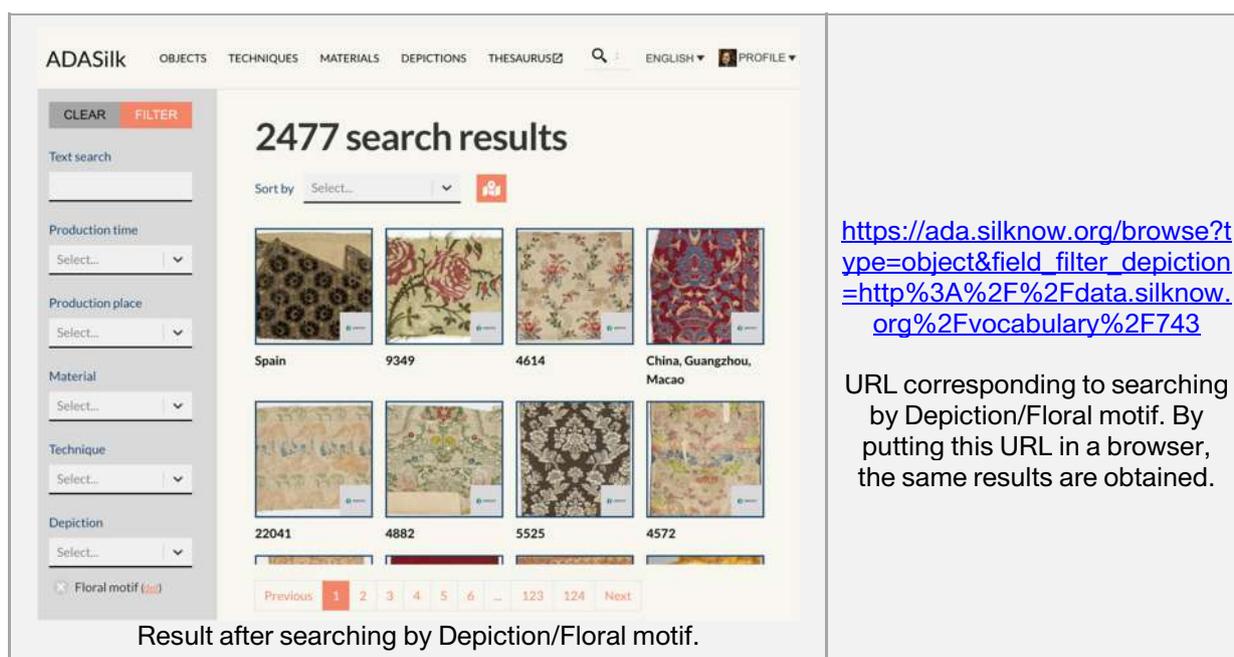


Figure 40. Screenshot and returned URL to test functionality 24.

3.2.25. Functionality 25

Description of the functionality: Providing the user with clear information about the licences / authors' rights applied to the images and what can and cannot be done with them.

Fulfilled: Yes.

Implementation/use: Information related to the authorship and/or rights applied to the images is available in the textual description of the object, when available.

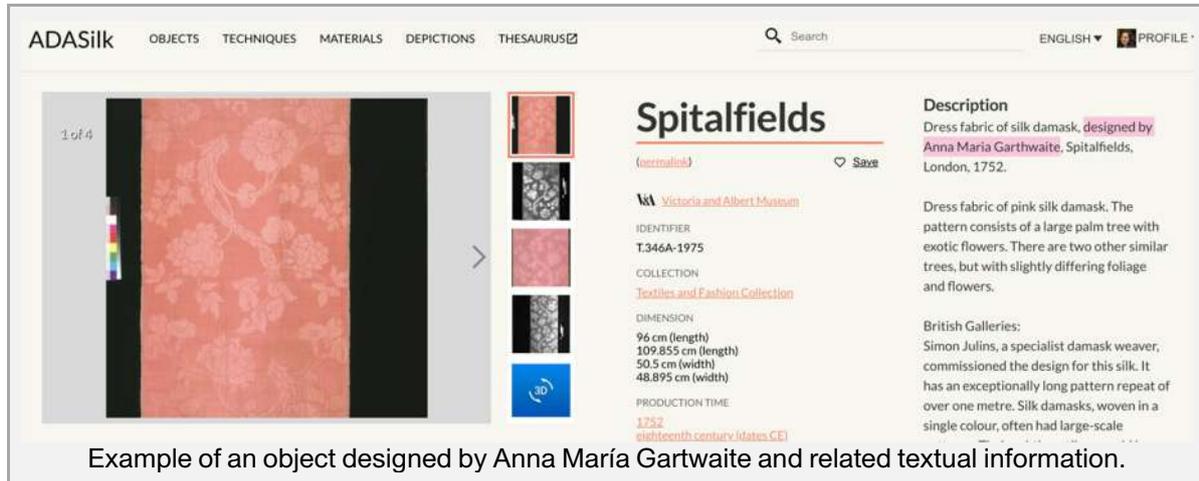


Figure 41. Screenshots of ADASilk to test functionality 25.

3.2.26. Functionality 26

Description of the functionality: Being able to contact the SILKNOW project.

Fulfilled: Yes.

Implementation/use: The email to contact SILKNOW's coordination team is available on all the pages in ADASilk. This can be seen in Figure 42

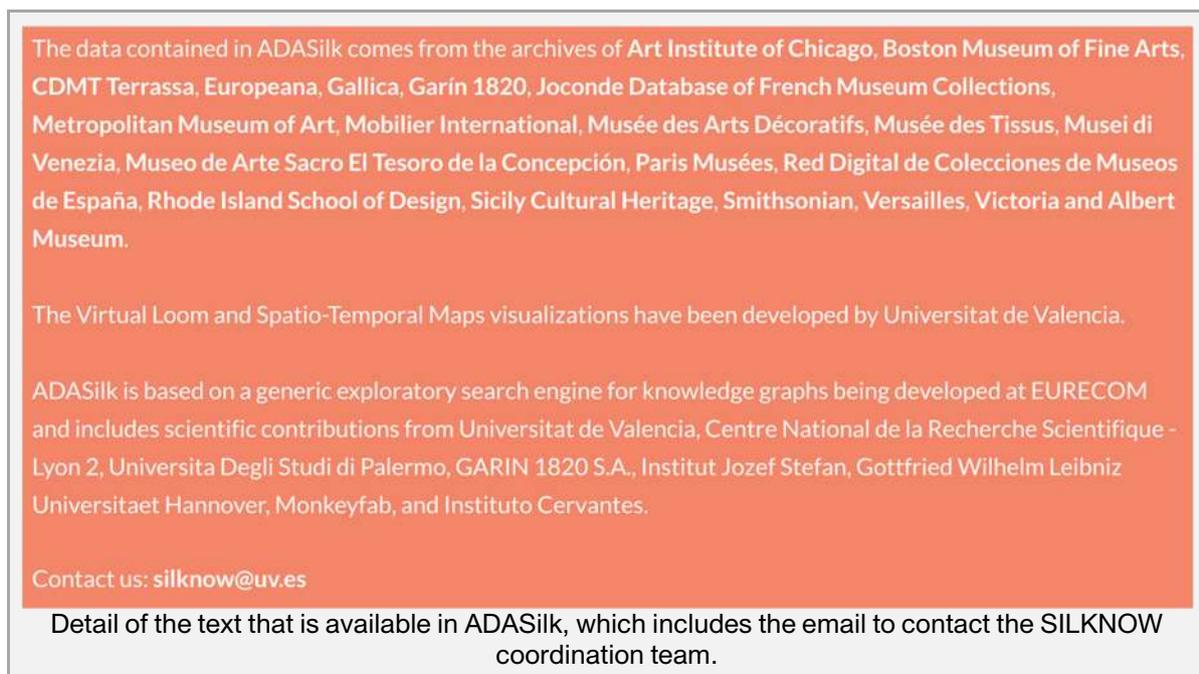


Figure 42. Screenshots of ADASilk to test functionality 26.

3.2.27. Discussion

The results provided by the Functional testing for ADASilk show very satisfactory outcomes for most of the functionalities which were identified by SSH partners in the scope of WP2. There are only two functionalities (numbers 04 and 18) which have not been fully implemented, although it is feasible to have them implemented by the end of the project. Therefore, the vast majority of the functionalities are already implemented and integrated into ADASilk's web application, which can be publicly accessed.

3.3. Stress testing

With the stress testing of ADASilk, we aim to determine its robustness in answering users' requests. ADASilk offers three different APIs to access the content of the SILKNOW Knowledge Graph:

- ADASilk API (Internal) used by the ADASilk web application
- SILKNOW's API (Public) powered by the SPARQL Transformer which can be used for third party integration
- SPARQL API powered by the triple store (in our deployment, Virtuoso)

The ADASILK APIs used to execute the stress test are described in deliverable D6.4.

To this end, we have prepared some experiments by simulating different numbers of concurrent users launching different requests, increasing the complexity of the query associated with the request. The experiments are described in section 3.3.1. During the experiments, we measured different parameters, explained in 3.3.2, which are used to assess if the tool performs well, even when producing and interacting with complex requests and a large number of concurrent users. The data gathered during the testing is provided in section 3.3.3 and analysed in sections 3.3.4 to 3.3.6. Finally, a brief discussion is set out in section 3.3.7.

3.3.1. Test description

We organized the stress tests according to the different SILKNOW Knowledge Graph access methods:

- We define a thread group made up of different numbers of concurrent users: 5, 10, 30 and 50.
- Each user process launches a batch of requests, separated by a random timer.
- All users' processes associated with the thread group execute their requests concurrently.
- The whole process is repeated five times (once all users complete a batch).

Table 8 depicts the structure of the thread groups and the requests performed for the stress test executed. This structure was repeated for each access method.

Number of users	Requests per user	Times repeated	Total requests
5	8	5	200
10	8	5	400

30	8	5	1200
50	8	5	2000

Table 8. Structure of the thread group of users and the number of requests performed for each stress test on one of the three access methods offered by SILKNOW.

The test configuration is the same as the one performed with the thesaurus stress test in order to clarify the process. Figure 10 shows a schema with the request execution process per request for three concurrent users.

Table 9 shows the timetable with the stress tests execution.

Time and date	Test launched
2021-06-25T04:00:00+0000	SILKNOW Public API (5 users)
2021-06-25T04:30:00+0000	ADASilk Internal API (5 users)
2021-06-25T05:00:00+0000	SPARQL API (5 users)
2021-06-25T06:00:00+0000	SILKNOW Public API (10 users)
2021-06-25T06:30:00+0000	ADASilk Internal API (10 users)
2021-06-25T07:00:00+0000	SPARQL API (10 users)
2021-06-24T10:00:00+0000	SILKNOW Public API (30 users)
2021-06-24T12:00:00+0000	ADASilk Internal API (30 users)
2021-06-24T14:00:00+0000	SPARQL API (30 users)
2021-06-24T16:00:00+0000	SILKNOW Public API (50 users)
2021-06-24T18:00:00+0000	ADASilk Internal API (50 users)
2021-06-24T20:00:00+0000	SPARQL API (50 users)

Table 9. Test stress execution timetable.

The hardware configuration which supports the three APIs services is described as follows:

- CPU: Intel Xeon L5640, 2.26 GHz, 12 cores (24 threads).
- RAM: 128 GB.
- Operating System: Linux Debian Buster, kernel 4.12.0.

The Knowledge Graph is hosted in a Virtuoso Docker replicated in a twin component. A load balancer between the two images is used to distribute the server load.

The client system where the tests were executed has the following characteristics:

- CPU: Intel i5-6400 CPU @ 2.70GHz.
- RAM: 8 GB.
- Operative System: Linux Fedora 7.0.

The tests were launched from the JMeter tool [4], using OpenJDK Java 1.8. We used this tool because of the API features and the tests' requirements, which makes Apache JMeter an adequate tool to perform such tests [5].

The tests are composed of a batch of requests. Each request has an associated query which is adapted to the request in order to be executed for the different evaluated APIs.

The queries are decomposed into two sets of four queries each. Inside a set, the queries have increased difficulty. The two different sets of queries, with different levels of difficulty, joined to the different and random timers per request execution, define a complex scenario which properly emulates a real situation.

The set of queries associated with the batch are:

Set 1:

- Production place: Italy,
- Text search: “damask”,
- Production time: eighteenth century (dates CE)
- Material: Metal thread

Set 2:

- Production Place: France
- Text search: “waistcoat”
- Technique: Velvet
- Material: silk thread

The specification of the adaptation of the queries to the different requests per API is detailed in ANNEX I of this document.

3.3.2. Definition of the parameters measured

In order to evaluate the performance of the APIs, given the different requests made, a lot of data are gathered per JMeter tool, but the parameters analyzed are:

- Elapsed time: the time elapsed between the time a user is issuing a request and a response is received.
- Fails: the requests can fail for various reasons indicated by the different error codes returned by the server (401, 404, 5, etc.).

The elapsed time is a very important parameter to define user experience. Based on the operation, if the elapsed time is longer than what is usually observed in other similar applications, the user experience is impacted.

Fail is the most critical situation, because the user must repeat the process in order to get the required data.

3.3.3. Gathered data during the tests

As mentioned above, in this section we show the data gathered during the tests that will be then analysed in subsections 3.3.4, 3.3.5 and 3.3.6. Table 10 gives a summary of the results of the tests, where the values presented are the average of the elapsed time per request and type of test, and the percentage of fails per request and type of test. Regarding the “fails percentage”, we measure if the response expected per request finished without errors. This is summarized in Table 10. Therefore, a result of “0” for this field means that all the response texts are received without errors.

In order to analyse the gathered data, we propose two graphics:

- A graphic with the average of the elapsed time per request and number of concurrent users.
- A graphic with the percentage of fails per request and number of concurrent users.

SILKNOW

This methodology is slightly different from the one used in the thesaurus analysis. In these tests, the number of requests is twice as important as in the previous tests, and mixing up all the data in one chart would have made the graphic look overloaded.

	CONCURRENT USERS	DATA	REQUEST								AVERAGE
			Italy	Italy+ damask	Italy+ damask + eighteen CE	Italy+ damask + eighteen CE+ metal thread	France	France+ waiscoat	France+ waistcoat + Velvet	France+ waistcoat + Velvet + silk thread	
ADASilk Public	5	Average Elapsed time	1650	6089	30060	1609	1789	11304	30046	1512	10507
		Fails percentage	0	0	100	0	0	0	100	0	25
	10	Average Elapsed time	1711	5422	28852	3521	2354	5895	29459	2165	9922
		Fails percentage	0	2	100	0	0	2	100	0	25.5
	30	Average Elapsed time	2046	13159	29998	8425	1636	14604	29837	19547	14906
		Fails percentage	20	28	100	66	30,6	17,3	100	70	54
50	Average Elapsed time	13003	13266	29705	12496	10901	9553	29709	12782	16427	
	Fails percentage	40	4.8	100	97.2	45.6	5.2	100	98.4	61.4	
ADASilk Internal	5	Average Elapsed time	401	214	271	234	310	240	245	30163	4010
		Fails percentage	0	4	4	0	0	0	0	100	13.5
	10	Average Elapsed time	280	509	727	3393	163	237	1853	30068	4645
		Fails percentage	0	0.57	1.14	13.71	0	0	5.71	100	15.11
	30	Average Elapsed time	463	315	297	307	313	267	277	28745	3873
		Fails percentage	0	2.6	13	8	4	4	2.6	100	16.75
50	Average Elapsed time	213	81	81	82	80	79	79	30007	3838	
	Fails percentage	0	0	0	0	0	0	0	100	12.5	
SPARQL	5	Average Elapsed time	184	55	93	8200	8377	136	174	208	2178
		Fails percentage	0	0	0	4	0	0	0	0	0.5
	10	Average Elapsed time	187	59	102	14457	10403	179	191	649	3278
		Fails percentage	0	2	0	50	34	0	0	0	10.75
	30	Average Elapsed time	177	51	95	12868	250	57	141	2777	2052
		Fails percentage	0	0	0	44.6	0	0	0	0.6	5.65
50	Average Elapsed time	245	61	97	13040	272	57	149	4889	2351	
	Fails percentage	0	0	1.2	43.2	0	0.8	0	4.8	6.25	

Table 10. Data gathered for the different access methods of the Knowledge Graph given the tests we have defined.

3.3.4. Results for ADASilk Internal API

Figure 43 shows the average elapsed time per request on the tests with 5, 10, 30 and 50 concurrent users, using the ADASilk Internal API.

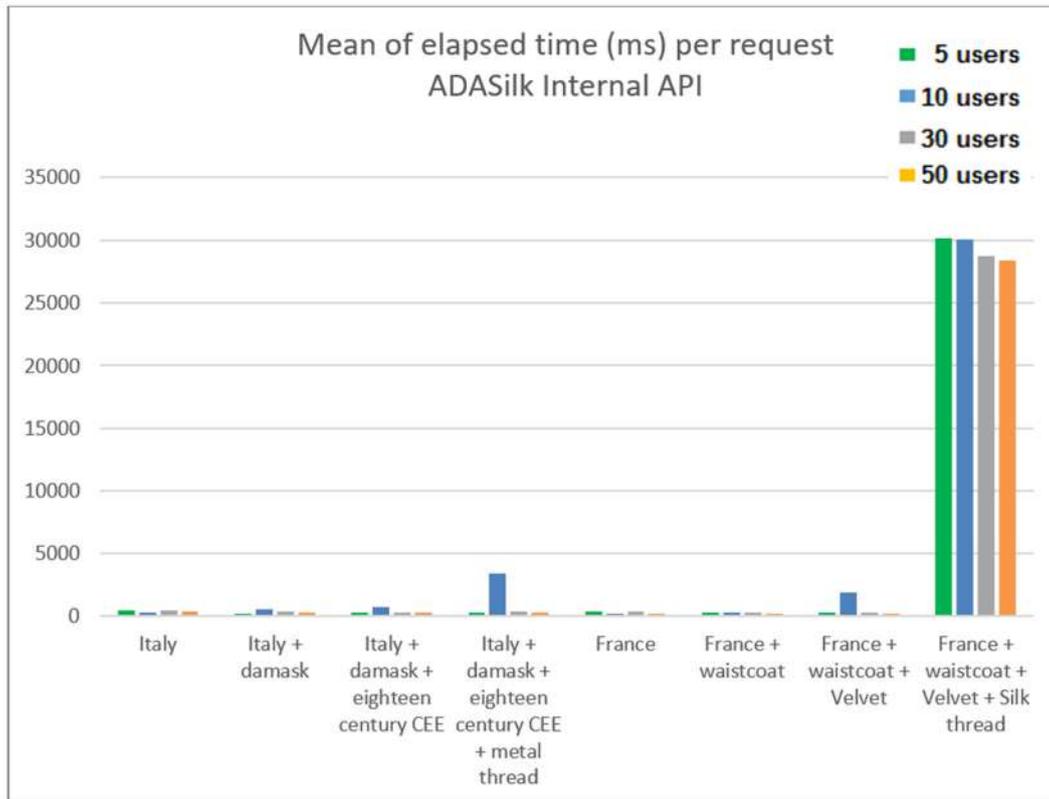


Figure 43. The mean of the elapsed time required per request in the tests performed with 5, 10, 30 and 50 concurrent users with the internal ADASilk API.

Figure 44 shows the percentage of fails per request on the tests with 5, 10, 30 and 50 concurrent users, using the ADASilk Internal API.

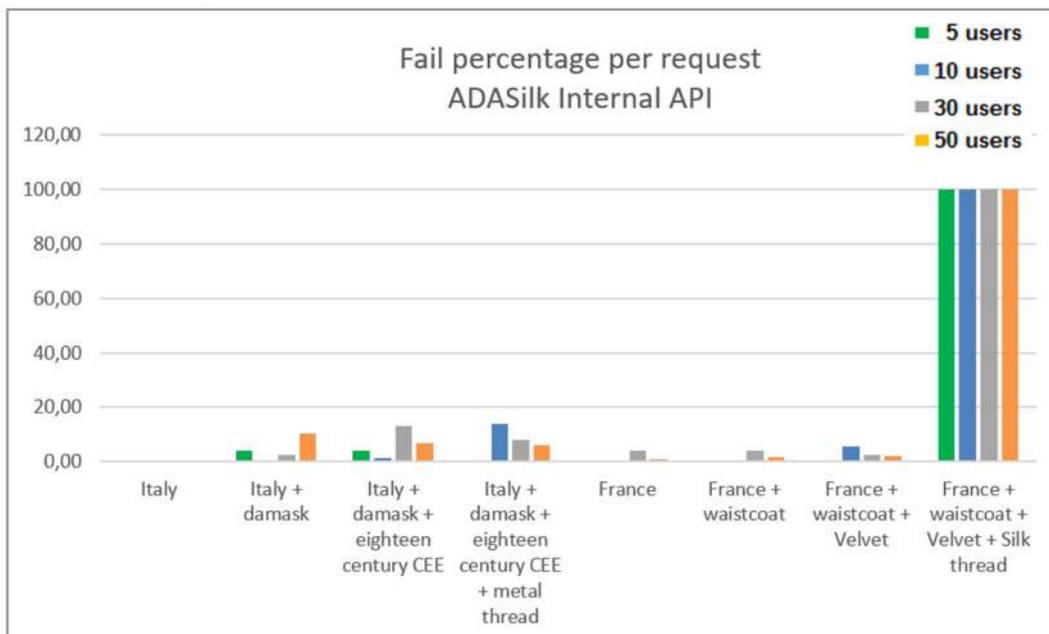


Figure 44. The percentage of fails per request in the tests performed with 5, 10, 30 and 50 concurrent users on the internal ADASilk API.

SILKNOW

If the last request is not taken into consideration, the stress tests related to the requests performed on the ADASILK Internal API have ended up with very good results on the elapsed time and in the number of fails per request. We conclude that with the current server configuration this API can manage up to 50 concurrent users.

The main problem is that there is one request which seems to always fail: the last request of the second set (the 8th) has a similar complexity to the 4th request in the first set. We have yet to discover what causes this discrepancy in the results.

On the other hand, the number of concurrent users does not seem to cause a specific problem for any of the tests carried out.

3.3.5. Results for SILKNOW Public API

Figure 45 shows the average elapsed time per request on the tests with 5, 10, 30 and 50 concurrent users with the SILKNOW Public API.

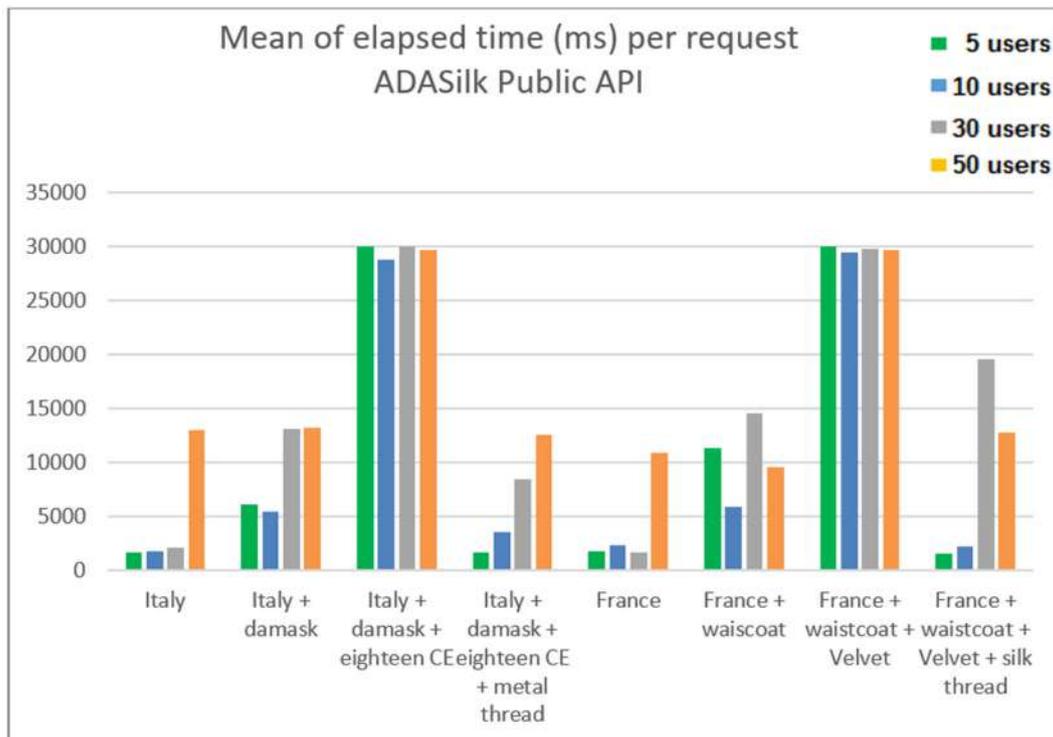


Figure 45. The mean of the elapsed time required per request in the tests performed with 5, 10, 30 and 50 concurrent users with the public SILKNOW API.

Figure 46 shows the percentage of fails per request on the tests with 5, 10, 30 and 50 concurrent users with the SILKNOW public API.

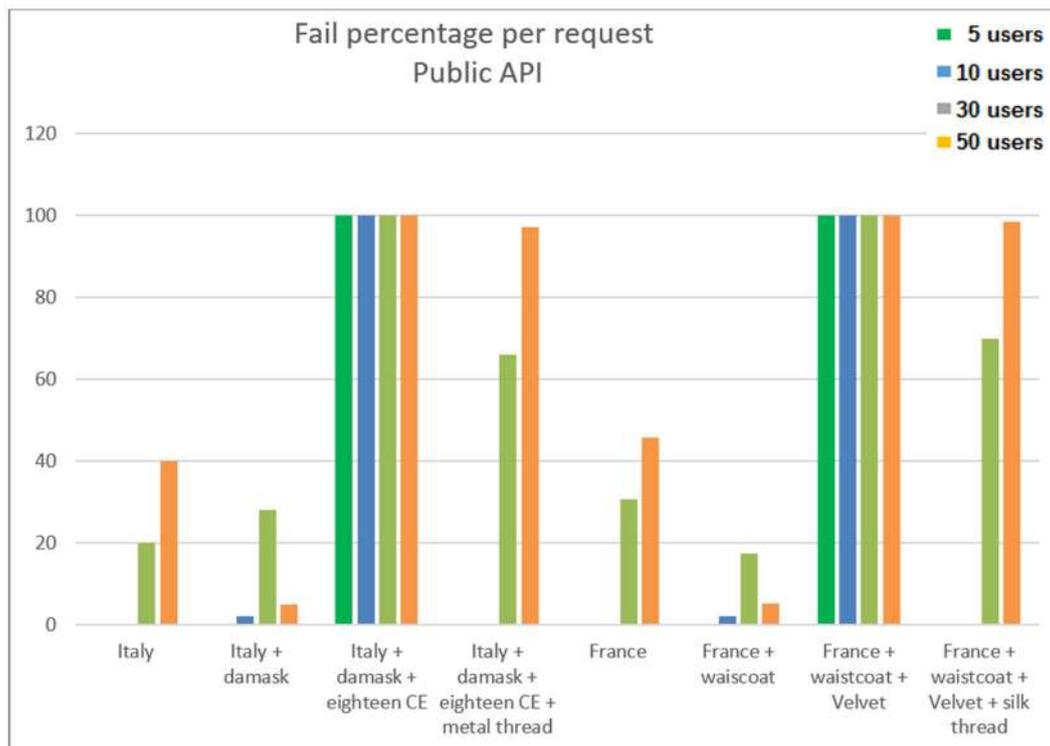


Figure 46. The percentage of fails per request in the tests performed with 5, 10, 30 and 50 concurrent users on the SILKNOW Public API.

The stress tests related to the requests performed on the SILKNOW Public API ended up with very good results on the elapsed time when the number of fails is not too large. The problem is that this number of fails is 100% in both the 3rd and the 7th request.

The 4th request has fewer fails than the 3rd one, but it is very high with tests executed with 30 concurrent users (greater than 60%) and the same situation occurs with tests executed with 50 concurrent users (almost 100%). This behaviour is abnormal since the query associated with the 4th request is more complex than the query associated with the 3rd request. We have yet to investigate why such a behaviour has been observed.

Given the results obtained, we recommend using this API with up to 10 concurrent users using this hardware configuration.

3.3.6. Results for SPARQL API

Figure 47 shows the average elapsed time per request on the tests executed with 5, 10, 30 and 50 concurrent users with the SPARQL API. Figure 48 shows the percentage of fails per request on the tests with 5, 10, 30 and 30 concurrent users with the SPARQL API.

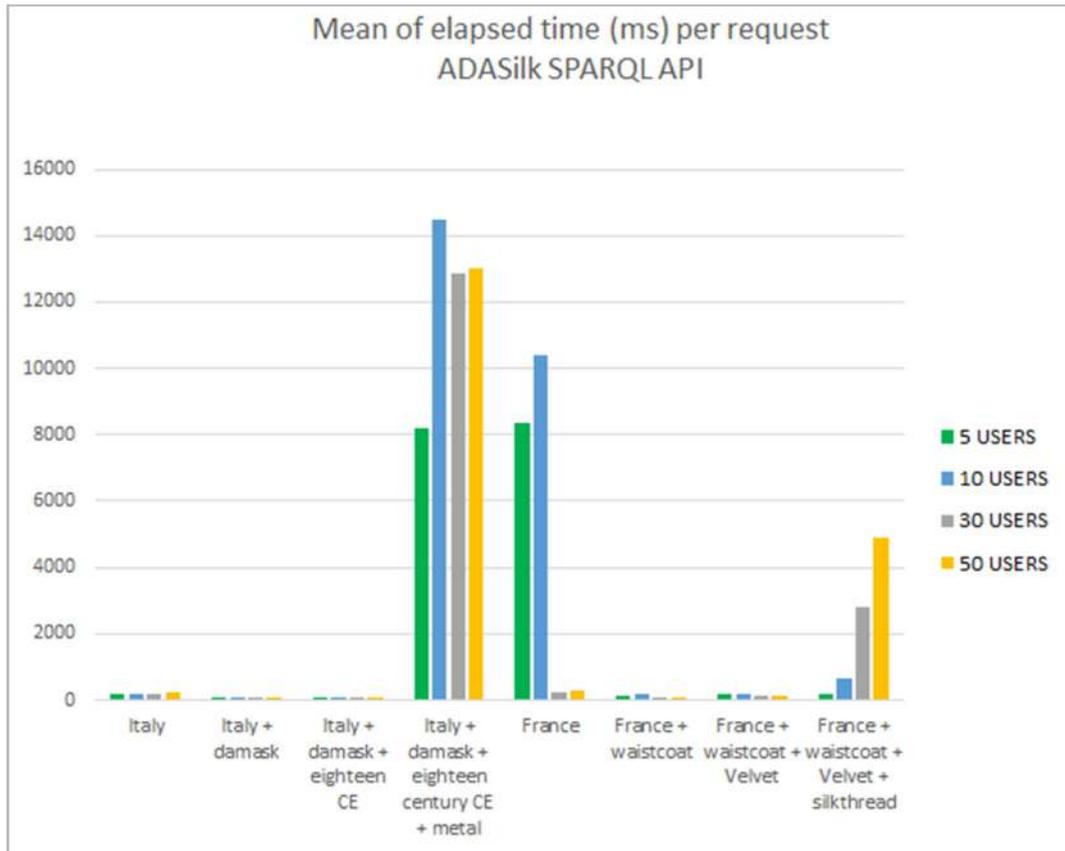


Figure 47. The mean of the elapsed time required per request in the tests performed with 5, 10, 30 and 50 concurrent users with the SPARQL API.

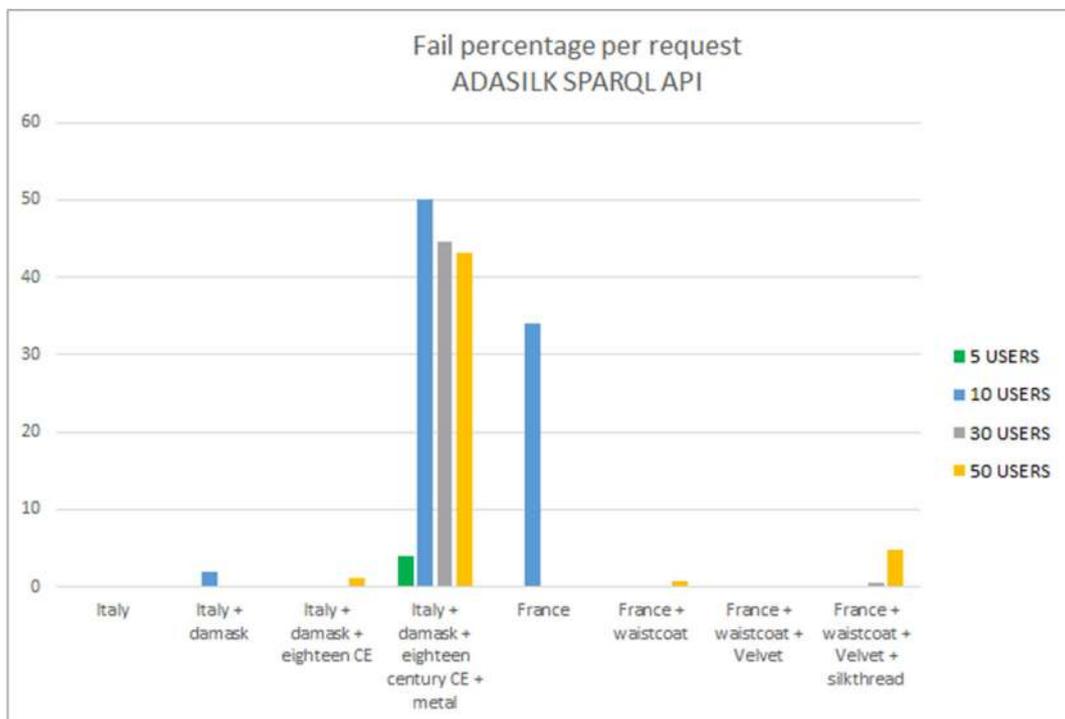


Figure 48. The percentage of fails per request in the tests performed with 5, 10, 30 and 50 concurrent users on the SPARQL API.

SILKNOW

The stress tests related to the requests performed on the SPARQL API ended up with very good results with regard to the elapsed time and the number of fails. The 4th request had a large number of fails in the tests executed with 10, 30 and 50 concurrent users (40%-50%) The 5th request also had a percentage of 35% of fails in the tests executed with 10 concurrent users. So, 4 requests had a significant number of fails, but always under 50% and only with tests executed with 10 or more concurrent users.

This API has the same problem with specific types of requests, but the problems are fewer than the ones with the SILKNOW Public API. In this case, the query never failed at a rate of 100%, and the number of requests affected is minimal.

3.3.7. Discussion

Given the hardware configuration, we conclude that the best performance is achieved using ADASilk. This is due, in part, to the smart caching we have implemented when developing this internal API. There is one specific query that fails, and it is necessary to find out why, in order to improve the system and to achieve an even better user experience.

The performance is lower using the other two access methods, since they do not have the smart caching. On the one hand, the SPARQL API is slightly more efficient than the SILKNOW API. On the other hand, the SPARQL API is harder to use for a Web client as the response is not web developer friendly. The difference between the two is the overhead provided by the SPARQL Transformer component. While we generally expect that this overhead is minor in most circumstances, we have yet to investigate why it can be so important for some queries.

4. SERVER STATISTICS

In order to add more data about the stress test, in this section the log of HAProxy is included, which we set up to control the 2 endpoints. It was active during the days the tests were executed and before, during the previous, and intermediate days. As a result of the 9 days of operations, the data of both endpoints is shown below:

- Endpoint 1: 372 522 requests. 94% resulting in 2xx (success) responses, 3% in 4xx (client errors) responses and 2% in 5xx (server errors) responses. The average response time is 15ms, while the maximum response time for a request was 49.8 seconds.
- Endpoint 2: 372 052 requests. 93% resulting in 2xx (success) responses, 3% in 4xx (client errors) responses and 2% in 5xx (server errors) responses. The average response time is 14ms, while the maximum response time for a request was 49.9 seconds.

The round robin worked well. Figure 49 shows the log summary of the HAProxy.

HAProxy

Statistics Report for pid 2164296

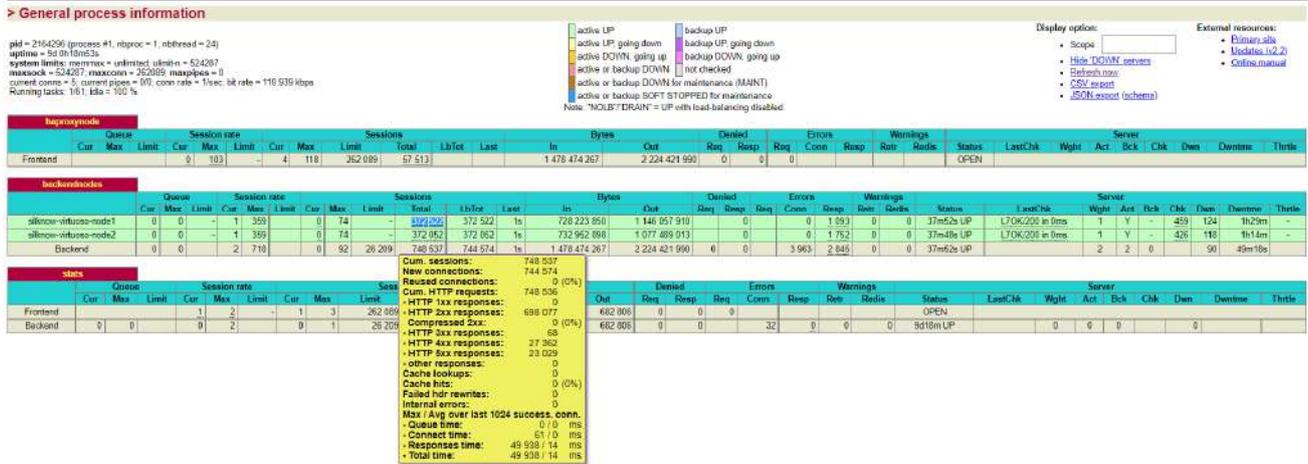


Figure 49. Summary of the logs of the HA Proxy with data concerning the two endpoints during the time the tests were executed.

5. CONCLUSIONS

In this deliverable, we have presented the result of the evaluation of two tools developed within the SILKNOW project, namely the SILKNOW thesaurus browser and the ADASilk web application.

Firstly, we evaluated the functionalities which we have developed. We conclude that the majority of functionalities have been fully developed, while the few remaining ones will be developed very soon.

Secondly, we have designed and executed stress tests for the different access methods to the SILKNOW Knowledge Graph and Thesaurus. When testing the access to the SILKNOW Thesaurus, we have noted that both the SILKNOW Public API and the SILKNOW SPARQL API provide excellent results, given the hardware configuration we have setup (i.e., a single server machine, with 2 Virtuoso replica hosting the same knowledge graph and a load balancer), for the different tests performed with 5, 10, 30 and 50 concurrent users. For third party integration, we recommend using the SILKNOW Public API built using the SPARQL Transformer, as it delivers more web developer friendly responses. The stress tests executed using the SKOSMOS software, which we have ourselves optimized, are good too, but not as good as the previous executed tests. The web interface adds a delay for specific actions, especially in the tests related to getting narrower and broader concepts. The user experience is acceptable with 10 concurrent users. We have yet to investigate how the SKOSMOS software could be further optimized to better handle this type of query.

When testing access to the SILKNOW Knowledge Graph, the best performance is attained using ADASilk. It uses an internal API that benefits from some smart caching, yielding better results in terms of average elapsed time and the percentage of fail requests. Some specific types of queries need to be investigated as they unexpectedly generate a higher percentage of fails. Considering the current hardware configuration, we conclude that the system can handle 50 concurrent users. The results of the other two other access methods (SILKNOW

SILKNOW

Public API and SPARQL API) are not so good. These are made for third party integration. Given the current hardware configuration, we recommend that they are suitable for 10 concurrent users. The difference between the two APIs, and the exact overhead provided by the SPARQL Transformer when implementing the SILKNOW Public API, has not been consistently measured and should be the topic of future research.

REFERENCES

- [1] P. Harping y M. Baca, Eds., *Introduction to Controlled Vocabularies: Terminology for Art, Architecture, and Other Cultural Works*. Los Angeles: The Getty Research Institute, 2015.
- [2] *NatLibFi/Skosmos*. National Library of Finland, 2021. Accedido: jul. 02, 2021. [En línea]. Disponible en: <https://github.com/NatLibFi/Skosmos>
- [3] «OpenLink Virtuoso SPARQL Query Editor». <https://data.silknow.org/sparql> (accedido jul. 02, 2021).
- [4] «Apache JMeter - Apache JMeter™». <https://jmeter.apache.org/> (accedido jul. 02, 2021).
- [5] R. Abbas, Z. Sultan, y S. N. Bhatti, «Comparative Study of Load Testing Tools: Apache JMeter, HP LoadRunner, Microsoft Visual Studio (TFS), Siege», *Sukkur IBA J. Comput. Math. Sci.*, vol. 1, n.º 2, Art. n.º 2, dic. 2017, doi: 10.30537/sjcms.v1i2.24.

ANNEX I. Request specification for stress test with ADASilk

Batch 1

Q1.1 Production Place: "Italy" (5901 results)

ADASilk API:

https://ada.silknow.org/api/search?field_filter_location=https%3A%2F%2Fsws.geonames.org%2F3175395%2F&page=1&type=object

SILKNOW API: [https://grlc.eurecom.fr/api-](https://grlc.eurecom.fr/api-git/silknow/api/obj_list?location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

[git/silknow/api/obj_list?location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql](https://grlc.eurecom.fr/api-git/silknow/api/obj_list?location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

SPARQL-Transformer (ADASilk):

```
{
  "@graph": [{
    "@id": "?id",
    "@graph": "?g"
  }],
  "$where": ["\n      GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production
ecrm:P108_has_produced ?id\n      .\n      ?production ecrm:P8_took_place_on_or_within
?location.OPTIONAL { ?location geonames:parentCountry ?parentCountry .
}\n      .\n      \n      \n      FILTER((?location = <https://sws.geonames.org/3175395/> ||
?parentCountry = <https://sws.geonames.org/3175395/>))\n      "],
  "$filter": [],
  "$offset": "0",
  "$limit": 20,
  "$prefixes": {
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "ecrm": "http://erlangen-crm.org/current/",
    "crmdig": "http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/",
    "dc": "http://purl.org/dc/elements/1.1/",
    "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
    "geonames": "http://www.geonames.org/ontology#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "silk": "http://data.silknow.org/ontology/property/"
  }
}
```

Generated SPARQL Query: [Link](#)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ecrm: <http://erlangen-crm.org/current/>
PREFIX crmdig: <http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX silk: <http://data.silknow.org/ontology/property/>
SELECT DISTINCT ?id ?g
WHERE {
  GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production ecrm:P108_has_produced ?id .
    ?production ecrm:P8_took_place_on_or_within ?location.OPTIONAL { ?location
geonames:parentCountry ?parentCountry . } .
  FILTER((?location = <https://sws.geonames.org/3175395/> || ?parentCountry =
<https://sws.geonames.org/3175395/>))
}
```

SILKNOW

LIMIT 20
OFFSET 0

Q1.2 Production Place: "Italy" & Text_Search: "damask" (3520 results)

ADASilk API:

https://ada.silknow.org/api/search?field_filter_location=https%3A%2F%2Fsws.geonames.org%2F3175395%2F&page=1&q=damask&type=object

SILKNOW API: [http://grlc.eurecom.fr/api-](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

[git/silknow/api/obj_list_text_search?text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

SPARQL-Transformer (ADASilk):

```
{
  "@graph": [{
    "@id": "?id",
    "@graph": "?g"
  }],
  "$where": ["\n      GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production
ecrm:P108_has_produced ?id\n      .\n      ?production ecrm:P8_took_place_on_or_within
?location.OPTIONAL { ?location geonames:parentCountry ?parentCountry .
}\n      .\n      \n      {\n      { ?id ?_s1p ?_s1o . ?_s1o bif:contains '\"damask*\"'
}\n      UNION\n      { ?id ?_s1p ?_s1o . FILTER(?_s1o = \"damask\")
}\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o bif:contains
 '\"damask*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . FILTER(?_s2o =
 \"damask\") }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o
 . ?_s3o bif:contains '\"damask*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p
 ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = \"damask\")
}\n      }\n      \n      .\n      FILTER((?location = <https://sws.geonames.org/3175395/> ||
?parentCountry = <https://sws.geonames.org/3175395/>))\n      "],
  "$filter": [],
  "$offset": "0",
  "$limit": 20,
  "$prefixes": {
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "ecrm": "http://erlangen-crm.org/current/",
    "crmdig": "http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/",
    "dc": "http://purl.org/dc/elements/1.1/",
    "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
    "geonames": "http://www.geonames.org/ontology#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "silk": "http://data.silknow.org/ontology/property/"
  }
}
```

Generated SPARQL Query: [Link](#)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ecrm: <http://erlangen-crm.org/current/>
PREFIX crmdig: <http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
```

SILKNOW

```
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX silk: <http://data.silknow.org/ontology/property/>
SELECT DISTINCT ?id ?g
WHERE {
  GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production ecrm:P108_has_produced ?id
  .
  ?production ecrm:P8_took_place_on_or_within ?location.OPTIONAL { ?location
  geonames:parentCountry ?parentCountry . }
  {
    { ?id ?_s1p ?_s1o . ?_s1o bif:contains '"damask*' ' ' }
    UNION
    { ?id ?_s1p ?_s1o . FILTER(?_s1o = "damask") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o bif:contains '"damask*' ' ' }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . FILTER(?_s2o = "damask") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . ?_s3o bif:contains '"damask*' ' ' }
  }
  UNION
  { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = "damask") }
  }
  .
  FILTER((?location = <https://sws.geonames.org/3175395/> || ?parentCountry =
  <https://sws.geonames.org/3175395/>))
}
LIMIT 20
OFFSET 0
```

Q1.3 Production Place: “Italy” & Text_Search: “damask” & Production_Time: eighteenth century (dates CE) (832 results)

ADASilk API:

https://ada.silknow.org/api/search?field_filter_location=https%3A%2F%2Fsws.geonames.org%2F3175395%2F&field_filter_time=http%3A%2F%2Fvocab.getty.edu%2Faat%2F300404512&page=1&q=damask&type=object

SILKNOW API: [http://grlc.eurecom.fr/api-](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?time=eighteenth%20century%20(dates%20CE)&text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

[git/silknow/api/obj_list_text_search?time=eighteenth%20century%20\(dates%20CE\)&text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?time=eighteenth%20century%20(dates%20CE)&text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

SPARQL-Transformer (ADASilk):

SILKNOW

```
{
  "@graph": [{
    "@id": "?id",
    "@graph": "?g"
  }],
  "$where": ["\n      GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production
ecrm:P108_has_produced ?id\n      .\n      ?production ecrm:P4_has_time-span ?time.OPTIONAL {
?time ecrm:P86_falls_within ?fallsWithin . }.?production ecrm:P8_took_place_on_or_within
?location.OPTIONAL { ?location geonames:parentCountry ?parentCountry .
}\n      .\n      \n      {\n      { ?id ?_s1p ?_s1o . ?_s1o bif:contains '\"damask*\"'
}\n      UNION\n      { ?id ?_s1p ?_s1o . FILTER(?_s1o = \"damask\")
}\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o bif:contains
 '\"damask*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . FILTER(?_s2o =
 \"damask\") }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o
 . ?_s3o bif:contains '\"damask*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p
 ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = \"damask\")
}\n      }\n      \n      .\n      FILTER((?time = <http://vocab.getty.edu/aat/300404512> ||
?fallsWithin = <http://vocab.getty.edu/aat/300404512>) && (?location =
<https://sws.geonames.org/3175395/> || ?parentCountry =
<https://sws.geonames.org/3175395/>))\n      "],
  "$filter": [],
  "$offset": "0",
  "$limit": 20,
  "$prefixes": {
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "ecrm": "http://erlangen-crm.org/current/",
    "crmdig": "http://www.ics.forth.gr/is1/CRMext/CRMdig.rdfs/",
    "dc": "http://purl.org/dc/elements/1.1/",
    "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
    "geonames": "http://www.geonames.org/ontology#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "silk": "http://data.silknow.org/ontology/property/"
  }
}
```

Generated SPARQL Query: [Link](#)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ecrm: <http://erlangen-crm.org/current/>
PREFIX crmdig: <http://www.ics.forth.gr/is1/CRMext/CRMdig.rdfs/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX silk: <http://data.silknow.org/ontology/property/>
SELECT DISTINCT ?id ?g
WHERE {
  GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production ecrm:P108_has_produced ?id
  .
  ?production ecrm:P4_has_time-span ?time.OPTIONAL { ?time ecrm:P86_falls_within ?fallsWithin
  . }.?production ecrm:P8_took_place_on_or_within ?location.OPTIONAL { ?location
  geonames:parentCountry ?parentCountry . }
  .
  {
    { ?id ?_s1p ?_s1o . ?_s1o bif:contains '"damask*' }
    UNION
    { ?id ?_s1p ?_s1o . FILTER(?_s1o = "damask") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o bif:contains '"damask*' }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . FILTER(?_s2o = "damask") }
  }
}
```

SILKNOW

```
UNION
{ ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . ?_s3o bif:contains '"damask*"'
}
UNION
{ ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = "damask") }
}
.
FILTER((?time = <http://vocab.getty.edu/aat/300404512> || ?fallsWithin =
<http://vocab.getty.edu/aat/300404512>) && (?location = <https://sws.geonames.org/3175395/> ||
?parentCountry = <https://sws.geonames.org/3175395/>))
}
LIMIT 20
OFFSET 0
```

Q1.4 Production Place: “Italy” & Text_Search: “damask” & Production_Time: eighteenth century (dates CE) & Material: Metal thread (55 results)

ADASilk API:

https://ada.silknow.org/api/search?&field_filter_material=http%3A%2F%2Fdata.silknow.org%2Fvocab%2F497&field_filter_location=https%3A%2F%2Fsws.geonames.org%2F3175395%2F&field_filter_time=http%3A%2F%2Fvocab.getty.edu%2Faat%2F300404512&page=1&q=damask&type=object

SILKNOW API: [http://grlc.eurecom.fr/api-](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?material=metal%20thread&time=eighteenth%20century%20(dates%20CE)&text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

[git/silknow/api/obj_list_text_search?material=metal%20thread&time=eighteenth%20century%20\(dates%20CE\)&text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?material=metal%20thread&time=eighteenth%20century%20(dates%20CE)&text=damask&location=Italy&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

SPARQL-Transformer (ADASilk):

```
{
  "@graph": [{
    "@id": "?id",
    "@graph": "?g"
  }],
  "$where": ["\n      GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production
ecrm:P108_has_produced ?id\n      .\n      ?production ecrm:P8_took_place_on_or_within
?location.OPTIONAL { ?location geonames:parentCountry ?parentCountry . }.?production
ecrm:P126_employed ?material.OPTIONAL { ?broaderMaterial (skos:member|skos:narrower)* ?material
}.?production ecrm:P32_used_general_technique ?technique.OPTIONAL { ?broaderTechnique
(skos:member|skos:narrower)* ?technique }\n      .\n      \n      {\n      { ?id ?_s1p ?_s1o .
?_s1o bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?id ?_s1p ?_s1o . FILTER(?_s1o =
\"waistcoat\") }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o
bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o .
FILTER(?_s2o = \"waistcoat\") }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o .
?_s2o ?_s3p ?_s3o . ?_s3o bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?_s1o ?_s1p
?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = \"waistcoat\")
}\n      }\n      \n      .\n      FILTER((?location = <https://sws.geonames.org/3175395/> ||
?parentCountry = <https://sws.geonames.org/3175395/>) && (?material =
<http://data.silknow.org/vocabulary/277> || ?broaderMaterial =
<http://data.silknow.org/vocabulary/277>) && (?technique =
<http://data.silknow.org/vocabulary/379> || ?broaderTechnique =
<http://data.silknow.org/vocabulary/379>))\n      "],
  "$filter": [],
  "$offset": "0",
  "$limit": 20,
  "$prefixes": {
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/"
  }
}
```

SILKNOW

```
"ecrm": "http://erlangen-crm.org/current/",
"crmdig": "http://www.ics.forth.gr/is1/CRMext/CRMdig.rdfs/",
"dc": "http://purl.org/dc/elements/1.1/",
"geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
"geonames": "http://www.geonames.org/ontology#",
"skos": "http://www.w3.org/2004/02/skos/core#",
"silk": "http://data.silknow.org/ontology/property/"
}
}
```

Generated SPARQL Query: [Link](#)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ecrm: <http://erlangen-crm.org/current/>
PREFIX crmdig: <http://www.ics.forth.gr/is1/CRMext/CRMdig.rdfs/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX silk: <http://data.silknow.org/ontology/property/>
SELECT DISTINCT ?id ?g
WHERE {
  GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production ecrm:P108_has_produced ?id
  .
  ?production ecrm:P4_has_time-span ?time.OPTIONAL { ?time ecrm:P86_falls_within ?fallsWithin
  . }.?production ecrm:P8_took_place_on_or_within ?location.OPTIONAL { ?location
  geonames:parentCountry ?parentCountry . }.?production ecrm:P126_employed ?material.OPTIONAL {
  ?broaderMaterial (skos:member|skos:narrower)* ?material }
  .
  {
    { ?id ?_s1p ?_s1o . ?_s1o bif:contains '"damask*' }
    UNION
    { ?id ?_s1p ?_s1o . FILTER(?_s1o = "damask") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o bif:contains '"damask*' }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . FILTER(?_s2o = "damask") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . ?_s3o bif:contains '"damask*' }
  }
  UNION
  { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = "damask") }
  }
  .
  FILTER((?time = <http://vocab.getty.edu/aat/300404512> || ?fallsWithin =
  <http://vocab.getty.edu/aat/300404512>) && (?location = <https://sws.geonames.org/3175395/> ||
  ?parentCountry = <https://sws.geonames.org/3175395/>) && (?material =
  <http://data.silknow.org/vocabulary/497> || ?broaderMaterial =
  <http://data.silknow.org/vocabulary/497>))
}
LIMIT 20
OFFSET 0
```

SILKNOW

Batch 2

Q2.1 Production Place: "France" (7079 results)

ADASilk API:

https://ada.silkknow.org/api/search?field_filter_location=https%3A%2F%2Fsws.geonames.org%2F3017382%2F&page=1&type=object

SILKNOW API: [https://grlc.eurecom.fr/api-](https://grlc.eurecom.fr/api-git/silkknow/api/obj_list?location=France&endpoint=http%3A%2F%2Fdata.silkknow.org%2Fsparql)

[git/silkknow/api/obj_list?location=France&endpoint=http%3A%2F%2Fdata.silkknow.org%2Fsparql](https://grlc.eurecom.fr/api-git/silkknow/api/obj_list?location=France&endpoint=http%3A%2F%2Fdata.silkknow.org%2Fsparql)

SPARQL-Transformer (ADASilk):

```
{
  "@graph": [{
    "@id": "?id",
    "@graph": "?g"
  }],
  "$where": ["\n      GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production
ecrm:P108_has_produced ?id\n      .\n      ?production ecrm:P8_took_place_on_or_within
?location.OPTIONAL { ?location geonames:parentCountry ?parentCountry .
}\n      .\n      \n      \n      FILTER((?location = <https://sws.geonames.org/3017382/> ||
?parentCountry = <https://sws.geonames.org/3017382/>))\n      ",
  "$filter": [],
  "$offset": "0",
  "$limit": 20,
  "$prefixes": {
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "ecrm": "http://erlangen-crm.org/current/",
    "crmdig": "http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/",
    "dc": "http://purl.org/dc/elements/1.1/",
    "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
    "geonames": "http://www.geonames.org/ontology#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "silk": "http://data.silkknow.org/ontology/property/"
  }
}
```

Generated SPARQL Query: [Link](#)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ecrm: <http://erlangen-crm.org/current/>
PREFIX crmdig: <http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX silk: <http://data.silkknow.org/ontology/property/>
SELECT DISTINCT ?id ?g
WHERE {
  GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production ecrm:P108_has_produced ?id
  .
  ?production ecrm:P8_took_place_on_or_within ?location.OPTIONAL { ?location
geonames:parentCountry ?parentCountry . }
  .
  FILTER((?location = <https://sws.geonames.org/3017382/> || ?parentCountry =
<https://sws.geonames.org/3017382/>))
}
```

SILKNOW

LIMIT 20
OFFSET 0

Q2.2 Production Place: “France” & text_search: “waistcoat” (3416 results)

ADASilk API:

https://ada.silknow.org/api/search?field_filter_location=https%3A%2F%2Fsws.geonames.org%2F3017382%2F&page=1&q=waistcoat&type=object

SILKNOW API: [http://grlc.eurecom.fr/api-](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?text=waistcoat&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

[git/silknow/api/obj_list_text_search?text=waistcoat&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?text=waistcoat&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

SPARQL-Transformer (ADASilk):

```
{
  "@graph": [{
    "@id": "?id",
    "@graph": "?g"
  }],
  "$where": ["\n      GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production
ecrm:P108_has_produced ?id\n      .\n      ?production ecrm:P8_took_place_on_or_within
?location.OPTIONAL { ?location geonames:parentCountry ?parentCountry .
}\n      .\n      \n      {\n      { ?id ?_s1p ?_s1o . ?_s1o bif:contains '\"waistcoat*\"'
}\n      UNION\n      { ?id ?_s1p ?_s1o . FILTER(?_s1o = \"waistcoat\")
}\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o bif:contains
 '\"waistcoat*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . FILTER(?_s2o
 = \"waistcoat\") }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p
 ?_s3o . ?_s3o bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o
 ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = \"waistcoat\")
}\n      }\n      \n      .\n      \n      FILTER((?location = <https://sws.geonames.org/3017382/> ||
?parentCountry = <https://sws.geonames.org/3017382/>))\n      "],
  "$filter": [],
  "$offset": "0",
  "$limit": 20,
  "$prefixes": {
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "ecrm": "http://erlangen-crm.org/current/",
    "crmdig": "http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/",
    "dc": "http://purl.org/dc/elements/1.1/",
    "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
    "geonames": "http://www.geonames.org/ontology#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "silk": "http://data.silknow.org/ontology/property/"
  }
}
```

Generated SPARQL Query: [Link](#)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ecrm: <http://erlangen-crm.org/current/>
PREFIX crmdig: <http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX silk: <http://data.silknow.org/ontology/property/>
SELECT DISTINCT ?id ?g
```

SILKNOW

```
WHERE {
  GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production ecrm:P108_has_produced ?id
  .
  ?production ecrm:P8_took_place_on_or_within ?location.OPTIONAL { ?location
  geonames:parentCountry ?parentCountry . }
  .
  {
    { ?id ?_s1p ?_s1o . ?_s1o bif:contains '"waistcoat*"' }
    UNION
    { ?id ?_s1p ?_s1o . FILTER(?_s1o = "waistcoat") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o bif:contains '"waistcoat*"' }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . FILTER(?_s2o = "waistcoat") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . ?_s3o bif:contains
    '"waistcoat*"' }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = "waistcoat") }
  }
  .
  FILTER((?location = <https://sws.geonames.org/3017382/> || ?parentCountry =
  <https://sws.geonames.org/3017382/>))
}
LIMIT 20
OFFSET 0
```

Q2.3 Production Place: “France” & text_search: “waistcoat”& Technique:”Velvet” (249 results)

ADASilk API:

https://ada.silknow.org/api/search?field_filter_location=https%3A%2F%2Fsws.geonames.org%2F3017382%2F&field_filter_technique=http%3A%2F%2Fdata.silknow.org%2Fvocabulary%2F379&page=1&q=waistcoat&type=object

SILKNOW API: [http://grlc.eurecom.fr/api-](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?text=waistcoat&technique=Velvet&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

[git/silknow/api/obj_list_text_search?text=waistcoat&technique=Velvet&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?text=waistcoat&technique=Velvet&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

SPARQL-Transformer (ADASilk):

```
{
  "@graph": [{
    "@id": "?id",
    "@graph": "?g"
  }],
  "$where": ["\n      GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production
  ecrm:P108_has_produced ?id\n      .\n      ?production ecrm:P8_took_place_on_or_within
  ?location.OPTIONAL { ?location geonames:parentCountry ?parentCountry . }.?production
  ecrm:P32_used_general_technique ?technique.OPTIONAL { ?broaderTechnique
  (skos:member|skos:narrower)* ?technique }\n      .\n      \n      {\n      { ?id ?_s1p ?_s1o .
  ?_s1o bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?id ?_s1p ?_s1o . FILTER(?_s1o =
  '\"waistcoat\"') }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o
  bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o .
  FILTER(?_s2o = '\"waistcoat\"') }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o .
  ?_s2o ?_s3p ?_s3o . ?_s3o bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?_s1o ?_s1p
  ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = '\"waistcoat\"')
  }\n      }\n      \n      \n      .\n      FILTER((?location = <https://sws.geonames.org/3017382/> ||
  ?parentCountry = <https://sws.geonames.org/3017382/>) && (?technique =
  <http://data.silknow.org/vocabulary/379> || ?broaderTechnique =
  <http://data.silknow.org/vocabulary/379>))\n      "],
```

SILKNOW

```
"$filter": [],
"$offset": "0",
"$limit": 20,
"$prefixes": {
  "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
  "schema": "http://schema.org/",
  "ecrm": "http://erlangen-crm.org/current/",
  "crmdig": "http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/",
  "dc": "http://purl.org/dc/elements/1.1/",
  "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
  "geonames": "http://www.geonames.org/ontology#",
  "skos": "http://www.w3.org/2004/02/skos/core#",
  "silk": "http://data.silknow.org/ontology/property/"
}
```

Generated SPARQL Query: [Link](#)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ecrm: <http://erlangen-crm.org/current/>
PREFIX crmdig: <http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX silk: <http://data.silknow.org/ontology/property/>
SELECT DISTINCT ?id ?g
WHERE {
  GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production ecrm:P108_has_produced ?id
  .
  ?production ecrm:P8_took_place_on_or_within ?location.OPTIONAL { ?location
geonames:parentCountry ?parentCountry . }.?production ecrm:P32_used_general_technique
?technique.OPTIONAL { ?broaderTechnique (skos:member|skos:narrower)* ?technique }
  .
  {
    { ?id ?_s1p ?_s1o . ?_s1o bif:contains '"waistcoat*' }
    UNION
    { ?id ?_s1p ?_s1o . FILTER(?_s1o = "waistcoat") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o bif:contains '"waistcoat*' }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . FILTER(?_s2o = "waistcoat") }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . ?_s3o bif:contains
'"waistcoat*' }
    UNION
    { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = "waistcoat") }
  }
  .
  FILTER((?location = <https://sws.geonames.org/3017382/> || ?parentCountry =
<https://sws.geonames.org/3017382/>) && (?technique = <http://data.silknow.org/vocabulary/379>
|| ?broaderTechnique = <http://data.silknow.org/vocabulary/379>))
}
LIMIT 20
OFFSET 0
```

Q2.4 Production Place: “France” & text_search: “waistcoat”& Technique:”Velvet” & Material:”silk thread” (11 results)

SILKNOW

ADASilk API:

https://ada.silknow.org/api/search?field_filter_location=https%3A%2F%2Fsws.geonames.org%2F3017382%2F&field_filter_material=http%3A%2F%2Fdata.silknow.org%2Fvocabulary%2F277&field_filter_technique=http%3A%2F%2Fdata.silknow.org%2Fvocabulary%2F379&page=1&q=waistcoat&type=object

SILKNOW API: [http://grlc.eurecom.fr/api-](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?material=silk%20thread&text=waistcoat&technique=Velvet&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

[git/silknow/api/obj_list_text_search?material=silk%20thread&text=waistcoat&technique=Velvet&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql](http://grlc.eurecom.fr/api-git/silknow/api/obj_list_text_search?material=silk%20thread&text=waistcoat&technique=Velvet&location=France&endpoint=http%3A%2F%2Fdata.silknow.org%2Fsparql)

SPARQL-Transformer (ADASilk):

```
{
  "@graph": [{
    "@id": "?id",
    "@graph": "?g"
  }],
  "$where": ["\n      GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production
ecrm:P108_has_produced ?id\n      .\n      ?production ecrm:P8_took_place_on_or_within
?location.OPTIONAL { ?location geonames:parentCountry ?parentCountry . }.?production
ecrm:P126_employed ?material.OPTIONAL { ?broaderMaterial (skos:member|skos:narrower)* ?material
}.?production ecrm:P32_used_general_technique ?technique.OPTIONAL { ?broaderTechnique
(skos:member|skos:narrower)* ?technique }\n      .\n      \n      {\n      { ?id ?_s1p ?_s1o .
?_s1o bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?id ?_s1p ?_s1o . FILTER(?_s1o =
\"waistcoat\") }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o . ?_s2o
bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o .
FILTER(?_s2o = \"waistcoat\") }\n      UNION\n      { ?_s1o ?_s1p ?id . ?_s1o ?_s2p ?_s2o .
?_s2o ?_s3p ?_s3o . ?_s3o bif:contains '\"waistcoat*\"' }\n      UNION\n      { ?_s1o ?_s1p
?id . ?_s1o ?_s2p ?_s2o . ?_s2o ?_s3p ?_s3o . FILTER(?_s3o = \"waistcoat\")
}\n      }\n      \n      .\n      FILTER((?location = <https://sws.geonames.org/3017382/> ||
?parentCountry = <https://sws.geonames.org/3017382/>) && (?material =
<http://data.silknow.org/vocabulary/277> || ?broaderMaterial =
<http://data.silknow.org/vocabulary/277>) && (?technique =
<http://data.silknow.org/vocabulary/379> || ?broaderTechnique =
<http://data.silknow.org/vocabulary/379>))\n      "],
  "$filter": [],
  "$offset": "0",
  "$limit": 20,
  "$prefixes": {
    "rdfs": "http://www.w3.org/2000/01/rdf-schema#",
    "schema": "http://schema.org/",
    "ecrm": "http://erlangen-crm.org/current/",
    "crmdig": "http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/",
    "dc": "http://purl.org/dc/elements/1.1/",
    "geo": "http://www.w3.org/2003/01/geo/wgs84_pos#",
    "geonames": "http://www.geonames.org/ontology#",
    "skos": "http://www.w3.org/2004/02/skos/core#",
    "silk": "http://data.silknow.org/ontology/property/"
  }
}
```

Generated SPARQL Query: [Link](#)

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ecrm: <http://erlangen-crm.org/current/>
PREFIX crmdig: <http://www.ics.forth.gr/isl/CRMext/CRMdig.rdfs/>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX geo: <http://www.w3.org/2003/01/geo/wgs84_pos#>
PREFIX geonames: <http://www.geonames.org/ontology#>
PREFIX skos: <http://www.w3.org/2004/02/skos/core#>
PREFIX silk: <http://data.silknow.org/ontology/property/>
```

SILKNOW

```
SELECT DISTINCT ?id ?g
WHERE {
  GRAPH ?g { ?id a ecrm:E22_Man-Made_Object }.?production ecrm:P108_has_produced ?id
  .
  ?production ecrm:P8_took_place_on_or_within ?location.OPTIONAL { ?location
geonames:parentCountry ?parentCountry . }.?production ecrm:P126_employed ?material.OPTIONAL {
?broaderMaterial (skos:member|skos:narrower) ?material }.?production
ecrm:P32_used_general_technique ?technique.OPTIONAL { ?broaderTechnique
(skos:member|skos:narrower) ?technique }

  FILTER((?location = <https://sws.geonames.org/3017382/> || ?parentCountry =
<https://sws.geonames.org/3017382/>) && (?material = <http://data.silknow.org/vocabulary/277> ||
?broaderMaterial = <http://data.silknow.org/vocabulary/277>) && (?technique =
<http://data.silknow.org/vocabulary/379> || ?broaderTechnique =
<http://data.silknow.org/vocabulary/379>))
}
LIMIT 20
OFFSET 0
```

